



PHILCO 2000  
**32KSYS**  
OPERATING SYSTEM

Technical Computer Services  
Engineering Staff  
May, 1968



## TABLE OF CONTENTS

	Page	
<b>SECTION I</b>	<b>GENERAL DESCRIPTION .....</b>	<b>1-1</b>
	An Aid to Operators.....	1-1
	An Aid to Programmers .....	1-1
	An Aid to Operations .....	1-1
	<b>General 32KSYS Operations .....</b>	<b>1-2</b>
	32KSYS Tape Assignment .....	1-2
	Communication with 32KSYS .....	1-2
	Control Lines .....	1-2
	32KSYS Entries .....	1-3
	Service Routines .....	1-3
	Library Routines .....	1-3
	<b>32KSYS Conventions.....</b>	<b>1-3</b>
	Memory Concept .....	1-3
	Sentinel Word .....	1-3
	Sentinel Block .....	1-4
	Optional Parameter .....	1-4
	Break Character .....	1-4
	Illegal Character .....	1-4
	Program Identity (D) .....	1-4
	Code Mode .....	1-4
	Hollerith-Code Mode .....	1-4
	Image Mode .....	1-4
	Hollerith-Image Mode .....	1-4
	Binary-Image Mode .....	1-5
	Address .....	1-5
	Dump .....	1-5
	Post Mortem Dump .....	1-5
	Snapshot Dump .....	1-5
	Symbolic Tape .....	1-5
	Preset Symbols in TAC .....	1-6
<b>SECTION II</b>	<b>SYMBOLIC TAPES .....</b>	<b>2-1</b>
	Introduction .....	2-1
	Symbolic Tape Names.....	2-1
	Operator Action .....	2-2
	Use of Symbolic Tape Names.....	2-2
	External Assignment .....	2-2
	Internal Assignment .....	2-3
	PUN'S Permitted.....	2-3
	SYS PUN'S .....	2-3
	Rule .....	2-3
<b>Tapes</b>	<b>2-4</b>	
<b>Release</b>	<b>2-5</b>	

	Page	
<b>SECTION III</b>	<b>SYSTEM TAPES .....</b>	<b>3-1</b>
<b>TSYSTEM</b>	- <b>System Program Tape.....</b>	<b>3-2</b>
<b>TSYSDMP</b>	- <b>System Scratch Tape .....</b>	<b>3-3</b>
<b>TSYSIN</b>	- <b>System Input Tape .....</b>	<b>3-4</b>
<b>TSYSOUT</b>	- <b>System Output Tape .....</b>	<b>3-5</b>
<b>TSYSLIB</b>	- <b>TAC Library Tape .....</b>	<b>3-8</b>
<b>TSYSCR0</b>	- <b>System Scratch Tape .....</b>	<b>3-9</b>
<b>TSYSCR3</b>	- <b>System Scratch Tape .....</b>	<b>3-10</b>
<b>TSTSCR4</b>	- <b>System Scratch Tape .....</b>	<b>3-11</b>
<b>TSYSCR6</b>	- <b>System Scratch Tape .....</b>	<b>3-12</b>
<b>SECTION IV</b>	<b>32KSYS XORD .....</b>	<b>4-1</b>
<b>Function</b>	.....	<b>4-1</b>
<b>Features</b>	.....	<b>4-1</b>
<b>Entrances to XORD</b>	.....	<b>4-2</b>
<b>Checking Orders</b>	.....	<b>4-2</b>
<b>Efficient Use of XORD</b>	.....	<b>4-2</b>
<b>MXORD</b>	- <b>Execute Order(s) .....</b>	<b>4-4</b>
The <b>MXORD Entrance</b>	.....	<b>4-4</b>
<b>Steps Taken in Issuing Orders</b>	.....	<b>4-5</b>
<b>Status of Registers on Return</b>	.....	<b>4-6</b>
<b>XORD</b>	- <b>Executes Order .....</b>	<b>4-7</b>
The <b>XORD Entrance</b>	.....	<b>4-7</b>
<b>CHKORD</b>	- <b>Checks an Order.....</b>	<b>4-8</b>
The <b>CHKORD Entrance</b>	.....	<b>4-8</b>
<b>CLORD</b>	- <b>Terminates an Order .....</b>	<b>4-9</b>
The <b>CLORD Entrance</b>	.....	<b>4-9</b>
<b>XTAPE</b>	- <b>Obtains Logical Tape Unit Number for PUN .....</b>	<b>4-10</b>
The <b>XTAPE Entrance</b>	.....	<b>4-10</b>
<b>ERROR EXITS AND RECOVERY PROCEDURES</b>	.....	<b>4-11</b>
<b>Error Typeout/Halt</b>	.....	<b>4-11</b>
<b>Error Messages</b>	.....	<b>4-11</b>
<b>SETXORD</b>	- <b>Establishes Error Exit .....</b>	<b>4-14</b>
The <b>SETXORD Entrance</b>	.....	<b>4-14</b>
<b>SPORD</b>	- <b>Established Error Exit for a Particular Order ...</b>	<b>4-15</b>
The <b>SPORD Entrance</b>	.....	<b>4-15</b>

	Page	
<b>SECTION V</b>	<b>CONTROL LINE FUNCTIONS .....</b>	<b>5-1</b>
Sources of Control Lines .....	5-1	
Control Line Format .....	5-1	
Remarks .....	5-3	
Writing Control Lines .....	5-4	
Control Line Functions .....	5-4	
<b>SECTION VI</b>	<b>INITIALIZATION FUNCTIONS .....</b>	<b>6-1</b>
JOB .....	6-2	
CONIN .....	6-4	
<b>SECTION VII</b>	<b>COMPILEATION FUNCTIONS .....</b>	<b>7-1</b>
TAC .....	7-2	
Console Typewriter Normal Type-Outs .....	7-4	
Console Typewriter Error Type-Outs .....	7-5	
ALTAC .....	7-6	
Console Typewriter Error Type-Outs .....	7-6	
ALTAC3X - A Fast Compile-and-Go Version of ALTACIII ..	7-7	
Console Typewriter Normal Type-Outs .....	7-8	
Console Typewriter Error Type-Outs .....	7-8	
Printer Error Indications .....	7-9	
COBOL .....	7-11	
Console Typewriter Error Type-Outs .....	7-11	
OPAL - The Assembly Language for Philco 1000 .....	7-12	
Console Typewriter Normal Type-Outs .....	7-12	
Console Typewriter Error Type-Outs .....	7-13	
REPORT GENERATOR .....	7-14	
Console Typewriter Error Type-Outs .....	7-14	
FORTRAN .....	7-15	
Console Typewriter Normal Type-Outs .....	7-17	
Console Typewriter Error Type-Outs .....	7-18	
<b>SECTION VIII</b>	<b>LOADING FUNCTIONS .....</b>	<b>8-1</b>
One Word Search.....	8-1	
GO-Executes Program(s) .....	8-2	
LOAD - Loads Binary Program Deck(s) .....	8-5	
ABS - ABS Loader Call .....	8-6	
REL - REL Loader Call .....	8-7	
RPL - RPL Loader Call .....	8-11	

	Page	
<b>SECTION IX</b>	<b>SEGMENTATION FUNCTIONS .....</b>	<b>9-1</b>
Segmentation Process .....	9-1	
Loading Segmented Programs .....	9-1	
<b>SEGMENT</b> - Defines a Program Segment .....	9-3	
<b>MASTER</b> - Defines a Master Segment .....	9-6	
<b>SECTION X</b>	<b>TAPE HANDLING FUNCTIONS .....</b>	<b>10-1</b>
<b>REWIND</b> - Rewinds Magnetic Tape .....	10-2	
<b>REWINDLO</b> - Rewinds Magnetic Tape with Lockout .....	10-3	
<b>READF</b> - Reads Magnetic Tape Forward .....	10-4	
<b>READB</b> - Reads Magnetic Tape Backward .....	10-6	
<b>WRITE</b> - Writes on Magnetic Tape .....	10-7	
<b>LOCSENT</b> - Locates a Sentinel on Magnetic Tape .....	10-8	
<b>LOCABS, LOCRLP, LOCTACL, LOCREL</b> - Locates a Program on Magnetic Tape .....	10-9	
<b>WRTABS</b> - Writes on Magnetic Tape in ABS Format .....	10-11	
<b>WRTRPL</b> - Writes on Magnetic Tape in RPL Format .....	10-14	
<b>WRTSENT</b> - Writes a Sentinel on Tape.....	10-17	
<b>SECTION XI</b>	<b>DEBUGGING FUNCTIONS .....</b>	<b>11-1</b>
<b>DUMP</b> - Dumps Memory .....	11-2	
<b>SNAP</b> - Snaps Memory .....	11-4	
<b>OCT</b> - Replaces Word(s) in Octal Format .....	11-8	
<b>CMD(or COMMAND)</b> - Replaces Program Instruction(s) in Command Format.....	11-10	
<b>ALPHA</b> - Replaces Word(s) in Alphanumeric Format .....	11-13	
<b>SECTION XII</b>	<b>SPECIAL FUNCTIONS .....</b>	<b>12-1</b>
<b>CLEAR</b> - Clears Memory .....	12-2	
<b>CLOCK</b> - Types Accounting Clock Date and Time on Console Typewriter .....	12-3	
<b>COMMON</b> - Sets Common Storage Area.....	12-4	
<b>ENDMEM</b> - Sets the Effective End of Memory .....	12-5	
<b>FILL</b> - Fills Memory .....	12-6	
<b>HLT</b> - Halts Computer Operation .....	12-7	
<b>IBIT</b> - Changes Indicator Bits .....	12-8	
<b>JMP</b> - Transfers Control.....	12-9	
<b>NOPRINT</b> - Inhibits Printing of Control Instructions .....	12-10	
<b>PRINT</b> - Permits Printing of Control Instructions.....	12-11	
<b>PROIG</b> - Sets the P-Relative Address in the COMMAND (CMD), OCT, and ALPHA Correction Routines .....	12-12	

	Page	
<b>PROGTAPE</b>	- Assign a Program Tape .....	12-13
<b>REM</b>	- Relays Remakrs to an Operator .....	12-14
<b>SYS<del>PS</del></b>	- Compiler Option Word .....	12-15
<b>SYMDEF</b>	- Defines a Symbolic Address .....	12-17
 <b>SECTION XIII</b>		
<b>1ADCONI</b>	32KSYS ENTRIES .....	13-1
	- Converts the First Parameter of a Control Instruction to an Octal Address.....	13-2
<b>1ADCONC</b>	- Converts the Next Parameter of a Control Instruction to an Octal Address.....	13-3
<b>1CLOCK</b>	- Obtains and Edits the Clock Word .....	13-4
<b>1ENCODE</b>	- Image to Code Conversion .....	13-5
<b>1ENDJOB</b>	- End of Job Routine .....	13-6
<b>1ERRDMP</b>	- Error Dump Routine.....	13-7
<b>1GETBL</b>	- Gets Blanks During Scanning .....	13-9
<b>1IGBL</b>	- Ignore Blanks During Scanning.....	13-10
<b>1INTCMD</b>	- Executes a 32KSYS Control Instruction .....	13-11
<b>1INTCON</b>	- Loads 1CONLIN + 1 to 1CONLIN + 9.....	13-12
<b>1INTLD</b>	- Internal Loader .....	13-13
<b>1MEMSIZ</b>	- Checks Memory Overflow .....	13-15
<b>1NXTCRD</b>	- Obtains the Next Card.....	13-16
<b>1NXTCON</b>	- Next Control Instruction Routine.....	13-17
<b>1PRTCON</b>	- Writes 1CONLIN on TSYSOUT .....	13-18
<b>1PRTMSG</b>	- Writes Message on TSYSOUT.....	13-19
<b>1PRTRUN</b>	- Empties Buffer for TSYSOUT .....	13-20
<b>1PUNSYM</b>	- Finds Symbolic Tape Name .....	13-21
<b>1SCAN</b>	- Scans the First Parameter of a Control Instruction Suppressing Spaces .....	13-22
<b>1SCANBL</b>	- Scans the First Parameter of a Control Instruction Recognizing Spaces .....	13-24
<b>1SCANON</b>	- Scans Additional Parameters of a Control Instruction .....	13-25
<b>1STRIPQ</b>	- Strips the Q Register of Spaces .....	13-29
<b>1SUBERR</b>	- Subroutine Error Exit.....	13-30
<b>1TAPENO</b>	- Obtains a Program Unit Number (PUN) .....	13-31
<b>1TYPOUT</b>	- Console Typewriter Type-Out .....	13-32
<b>1XCONER</b>	- Control Line Error Routine .....	13-33
	Error Messages .....	13-33
<b>1EXECUTI</b>	- Executes the Control Line in 1CONLIN + 1 to 1CONLIN + 9 .....	13-36
<b>1ZERO</b>	- Exponent Fault Exit .....	13-37

	Page	
<b>SECTION XIV</b>	<b>32KSYS SERVICE ROUTINES .....</b>	<b>14-1</b>
Calling Service Routines .....	14-1	
<b>AIDE SERVICE ROUTINE .....</b>	<b>14-2</b>	
Function .....	14-2	
Service Routine Initiation .....	14-2	
Input-Output System .....	14-2	
Control Instructions.....	14-2	
Copy .....	14-3	
COMPF .....	14-5	
COMPB .....	14-6	
SPACEF .....	14-8	
SPACEB .....	14-8	
ENDALL .....	14-8	
AIDE Example .....	14-9	
<b>ANALYZER SERVICE ROUTINE .....</b>	<b>14-10</b>	
Function .....	14-10	
Service Routine Initiation .....	14-10	
Program To Be Analyzed .....	14-10	
Input-Output System .....	14-10	
Control Instructions .....	14-10	
Analyzer Example.....	14-11	
Output Format .....	14-11	
<b>DATA SERVICE ROUTINE .....</b>	<b>14-13</b>	
Function .....	14-13	
Service Routine Initiation .....	14-13	
Input-Output System .....	14-13	
Control Instructions .....	14-13	
TAPE .....	14-13	
REMARKS .....	14-14	
ENDDATA .....	14-15	
ERRORS .....	14-15	
DATA Example .....	14-15	
<b>FLID SERVICE ROUTINE .....</b>	<b>14-17</b>	
Service Routine Initiation .....	14-17	
Input Requirements .....	14-17	
Input-Output System .....	14-17	
Control Instructions .....	14-17	
SENTINEL .....	14-18	
FIND .....	14-18	
LIST .....	14-19	
ENDALL .....	14-21	

	Page
<b>PLUM SERVICE ROUTINE .....</b>	<b>14-23</b>
Function .....	14-23
Service Routine Initiation .....	14-23
Input-Output System .....	14-23
Control Instructions .....	14-23
Positioning Items In The Library .....	14-24
OLDLIB .....	14-25
NEWLIB .....	14-26
GENTAPE .....	14-27
ADDGEN .....	14-28
ABSOLUTE .....	14-28
ENDGEN .....	14-29
ADDMACRO .....	14-30
ENDMACRO .....	14-31
ADDSUB .....	14-32
ENDSUB .....	14-32
ADDBRS .....	14-33
DELETE .....	14-35
ENDALL .....	14-35
Console Typewriter Normal Type-outs .....	14-36
Console Typewriter Error Type-outs .....	14-36
Error Printouts .....	14-37
<b>RPLC SERVICE ROUTING .....</b>	<b>14-42</b>
Function .....	14-42
Service Routine Initiation .....	14-42
Input-Output System .....	14-42
Control Instructions .....	14-42
NEWTAPE .....	14-43
COPY .....	14-43
COPYTIL .....	14-44
SKIPTIL .....	14-45
DELETE .....	14-45
ADD .....	14-45
CORRECT .....	14-45
END .....	14-48
IDCHANGE .....	14-48
INDEX .....	14-49
ENDALL .....	14-49
RPLC Service Routine Example .....	14-50
ERRORS .....	14-50

	<b>Page</b>
<b>TACSERV SERVICE ROUTINE .....</b>	<b>14-53</b>
Function .....	14-53
Service Routine Initiation .....	14-53
Input Requirements .....	14-53
Input-Output .....	14-53
Control Instructions .....	14-53
NEWTAPE .....	14-55
Class I Instructions .....	14-56
ADDPORG .....	14-56
COPY .....	14-57
COPYTIL .....	14-58
CORRECT .....	14-58
DELPORG .....	14-59
ENDALL .....	14-60
LOCATE .....	14-61
SENTINEL .....	14-62
Class II Instructions .....	14-62
Transfer of Data .....	14-62
ADD .....	14-63
DELADD .....	14-66
DELETE .....	14-68
ENDPROG .....	14-70
LOCFLAD .....	14-71
LOCSEQ .....	14-71
NOSEQ .....	14-72
REPLACE .....	14-72
SEQ .....	14-74
TACSERV Output .....	14-76
TALSERV Example .....	14-78
<b>TAPEDUMP SERVICE ROUTINE .....</b>	<b>14-80</b>
Service Routine Initiation .....	14-80
Input-Output System .....	14-80
Control Instructions .....	14-80
PROCESS .....	14-80
SPACEF .....	14-81
SPACEB .....	14-81

	Page
<b>SECTION XV                    APPLICATIONS .....</b>	<b>15-1</b>
PERT     - Performance Evaluation and Review Technique .....	15-2
STAT/2000 .....	15-4
XMAS     - Expandable Machine Accounting System .....	15-5
LP        - Linear Programming .....	15-6
<b>SECTION XVI                  TAC LIBRARY ROUTINES .....</b>	<b>16-1</b>
LOADGEN - Loading Generator .....	16-2
SNAPGEN - Snapshot Generator .....	16-4
<b>APPENDIX A                  32KSYS OPERATING PROCEDURES .....</b>	<b>A-1</b>
<b>APPENDIX B                  ALPHABETICAL LISTING OF CONSOLE TYPEWRITER TYPE-OUTS .....</b>	<b>B-1</b>

## **SECTION I**

### **GENERAL DESCRIPTION**

32KSYS, an operating system for Philco 2000 computers with 32,768 words of core storage, is an assembly of executive routines designed to provide efficient use of computer run time by standardizing operating procedures, and by reducing the necessity for operator intervention.

#### **AN AID TO OPERATORS**

32KSYS relieves the operator from manually performing such functions as program loading, rewinding tapes and enacting post-mortem dumps. It also minimizes the amount of tape mounting and dismounting between JOBS by providing a symbolic tape assignment facility.

#### **AN AID TO PROGRAMMERS**

32KSYS provides many specialized routines to aid the programmer in running and debugging programs. Moreover, 32KSYS Entries, or internal subroutines, may be used to reduce such programmer tasks as repeating numerous instructions to the operator, or calling external subroutines.

#### **AN AID TO OPERATIONS**

Additional time saving for computer centers is gained as 32KSYS can accept in succession any type of source language program (i.e., TAC or ALTAC) to be compiled and run. 32KSYS also provides memory dumps or snapshots in the event of program failure.

32KSYS reduces tape handling time by permitting input programs to be stacked (placed in consecutive order) on the same tape prior to compilation or running. Similarly, object programs (those which have been compiled) and edited outputs can be stacked on their respective output tapes.

## GENERAL 32KSYS OPERATIONS

32KSYS is on magnetic tape during the compilation or running of a program. Most of its operations are also contained within memory at all times to direct the operation of programs. As other sections of 32KSYS (not memory contained, because of infrequent use) are required, they are read as three-block-length routines into memory from tape and executed. 32KSYS requires 4096 words of memory (locations zero through  $7777_8$ ), leaving the remainder of the 32,768 words of memory to the programmer's use.

### 32KSYS TAPE ASSIGNMENT

32KSYS contains a symbolic tape-assignment facility (see Section II). Tapes may be mounted on any available transport. However, in order that 32KSYS operate at maximum efficiency, the system tape must be assigned unit 1, and the intermediate dump tape must be assigned unit 2.

### COMMUNICATION WITH 32KSYS

Communication between 32KSYS and the programmer may be made by Control Lines, 32KSYS Entries, 32KSYS Service Routines, Compilers, and TAC Library Routines for 32KSYS.

#### Control Lines

Control Lines are written by the programmer in standard TAC instruction format and are submitted from outside a user's program. Each control line consists of a control instruction and its parameters, and causes actions such as.

- reading writing, or rewinding tapes
- loading programs into memory and transferring control to them
- stopping computer action or transferring control to a location in memory
- signifying the types of debugging memory dumps desired.

**32KSYS Entries**

32KSYS Entries permit the programmer to call various routines stored within the basic operations of 32KSYS. The programmer may call these routines by writing a transfer of control from his program to a desired entry. Each entry is a location within 32KSYS which transfers control to its respective routine. The actual locations of these entries within 32KSYS are defined by preset symbol definitions in TAC.

The entries include provisions for such operations as:

- post-mortem dumps
- Console Typewriter type-outs
- reading, writing, and rewinding tapes
- scanning routines for the Address and Remarks field of a TAC-format card.

**Service Routines**

32KSYS Service Routines are generally called by writing a control line that designates the service routine as one of its parameters. These routines are specialized programs designed to aid the programmer in performing routine functions of tape maintenance, debugging, and program monitoring.

**Library Routines**

Library Routines include special subroutines and generators applicable to programs run under 32KSYS control. These are called by the programmer in the same manner as any TAC subroutine or generator.

## 32KSYS CONVENTIONS

The following conventions, definitions, and special 32KSYS symbols are used throughout the manual.

**MEMORY CONCEPT**

The beginning of memory is referred to as Location Zero, while the high-order locations are referred to as the end of memory.

**SENTINEL WORD**

8 or fewer alphanumeric characters, excluding break characters. Spaces are ignored. If fewer than eight characters are used, the sentinel word will be right justified and left-filled with zeros.

SENTINEL BLOCK	A block of 128 identical words. Only the first 120 words of the block are inspected by 32KSYS, however, during a sentinel search. Break characters are never included in the configuration of a sentinel block produced by the system.
OPTIONAL PARAMETER	Optional parameters must be indicated by the presence of a break character, unless otherwise noted
BREAK CHARACTER	One of four designated characters is used to separate parameters within 32KSYS control lines. It may be a comma (,), period (.), semicolon (;), or slash (/). No distinction is made between these break characters except where otherwise noted in this manual.
ILLEGAL CHARACTER	The special Philco character e (Hollerith code 7) is an illegal character when entered via a 32KSYS control line parameter
PROGRAM IDENTITY (ID)	Sixteen or fewer alphanumeric characters used as a unique configuration to label a program. Spaces are significant. Refer to One Word Search under Loading Functions
CODE MODE	The format resulting from the card-to-tape conversion of cards punched in Hollerith code into Philco 2000 Code by the Punched Card Controller. If Code Mode is indicated, 32KSYS assumes that data has been converted to Code Mode 10 words per card, 12 cards per block. (Refer to Philco Codes, Appendix B.)
HOLLERITH-CODE MODE	The same as Code Mode
IMAGE MODE	The format resulting from the card-to-tape process that transcribes the twelve rows of each column of a card as 12 binary digits onto tape. Punches are transcribed as ones. If Image Mode is indicated, 32 KSYS assumes that data has been transcribed in Image Mode, 20 words per card, six cards per block. Image Mode format may be in either Hollerith-Image Mode or Binary-Image Mode.
HOLLERITH-IMAGE MODE	The format resulting from Image Mode processing of cards punched in Hollerith code.

**BINARY-  
IMAGE MODE**

The format resulting from image Mode processing of cards punched in binary form. Binary cards may be read only in Image Mode.

**ADDRESS**

The symbols used in the description of control instruction parameters that call for an address, are indicative of the form in which they are to be supplied. Three symbols are used:

m an octal number from 10000 to 77777.

loc an octal number from 10000 to 77777; or an octal number from 0 to 67777 suffixed by a P, indicating the location is relative to a program origin; or an octal number from 0 to 67777 suffixed by a C indicating the location is relative to common.

addr an octal number from 10000 to 77777, or an octal number from 0 to 67777 suffixed by a P, indicating the location is relative to a program origin, or an octal number from 0 to 67777 suffixed by a C, indicating the location is relative to common, or a symbol in the form NAME,FLAD.

addr may be augmented by  $\pm n$  or  $\pm nh$ , where n is a decimal number.

**DUMP**

A dump is edited information from designated memory locations which is subsequently transmitted onto magnetic tape for off-line printing on the High-Speed Printer.

**POST-MORTEM  
DUMP**

A post-mortem dump is a dump which occurs when control is transferred to the 32KSYS entry 1ZERO, 1ERRDMP, or 1SUBERR. The dump is specified by the DUMP control instruction.

**SNAPSHOT DUMP**

A snapshot dump is a dump which occurs at designated locations during the running of an object program, as specified by the SNAP control instruction.

**SYMBOLIC TAPE**

The tape which is defined by a symbolic tape name (refer to Symbolic Tapes) or a system program unit number, or a program unit number which has been previously defined by a TAPES instruction.

**PRESET SYMBOLS  
IN TAC**

The following is a list of preset TAC symbols:

SYS.1ZERO	SYS.1PRTRUN
SYS.1SYSIN	SYS.1XORD
SYS.1ERRDMP	SYS.1SPORD
SYS.1ENDJOB	SYS.1SETXRD
SYS.1NXTCON	SYS.1CHKORD
SYS.1TYPOUT	SYS.1MXORD
SYS.1ENCODE	SYS.1COMPER
SYS.1CLOCK	SYS.1XTAPE
SYS.1MEMSIZ	SYS.1TRYJA
SYS.1SCANBL	SYS.1START
SYS.1GETBL	SYS.1CONBIT
SYS.1IGBL	SYS.1CHARCT
SYS.1SCAN	SYS.1BIT
SYS.1SCANON	SYS.1SYSOPS
SYS.1STRIPQ	SYS.1CONLIN
SYS.1XCONER	SYS.1DATE
SYS.1NXTCRD	SYS.1BRKSTR
SYS.1INTLTD	SYS.1WRD1
SYS.1SYSRET	SYS.1WRD2
SYS.1ADCONI	SYS.1CLORD
SYS.1ADCONC	XORD. XORD
SYS.1INTCMD	SPORD. SPORD
SYS.1TAPENO	SETXORD. SETXORD
SYS.1EXECUTI	CHKORD. CHKORD
SYS.1INTCON	MXORD. MXORD
SYS.1PUNSYM	XTAPE. XTape
SYS.1PRTCON	CLORD. CLORD
SYS.1PRTMSG	SYS.1MSGCON
SYS.1SYSHLT	

## SECTION II

### SYMBOLIC TAPES

#### INTRODUCTION

Tape assignment is that procedure whereby a particular reel of tape is connected in some manner with a particular tape unit in preparation for input-output processing of the reel. In general, there are two methods of assigning tapes for a job.

The first method requires the programmer to pre-determine the connection of reels and units and to specify this connection to the operator. This method has proven to be prohibitive because often two independent successive jobs require different file tapes on the same logical tape unit. To prevent the delay associated with rewinding and dismounting the previous file tape and then mounting the required file tape, operators usually mount the second file tape on a different (unused) unit and when the file tape is needed, replug the logical/physical connection.

The second method enables the operator to mount the file tape on any unit and to communicate the assignment to the program. This is the technique implemented in 32KSYS. It requires little, if any, replugging of logical/physical connections to provide tape assignment. It does require the use of symbolic tape names.

#### SYMBOLIC TAPE NAMES

A symbolic tape name is a collection of seven or fewer characters, one of which must be non-numeric, which uniquely identifies a tape reel for a particular job. Several special character sets are excluded from use within an object program.

These are the System symbolic tape names:

TSYSTEM	TSYSCR0
TSYSDMP	TSYSCR3
TSYSOUT	TSYSCR4
TSYSLIB	TSYSCR6
TSYSIN	

## OPERATOR ACTION

At some point during the running of each job the operator must communicate to 32KSYS the logical unit number of each significant reel of tape. (A detailed description of operator action is given in *Operating Procedures, Appendix A.*)

## USE OF SYMBOLIC TAPE NAMES

Symbolic tape names should be used whenever a tape parameter is specified, either in a system control instruction, or in an object program. Since the 32KSYS input-output processor, XORD, requires a number (PUN, or program unit number) in every order specifying the reel to be processed, symbolic tape names must be assigned to PUN's.

There are two methods of assigning a symbolic name to a PUN external and internal. The external method implies that the PUN ultimately appearing in the input-output order has been selected externally, and has not been predetermined by the programmer. The internal method implies that the PUN appearing in the order has been predetermined by the programmer.

## EXTERNAL ASSIGNMENT

The external method requires that the object program perform certain functions during its execution. These are:

- Given a symbolic name for a tape, ask for a PUN to be associated with this name via the SYS subroutine I TAPENO,
- Insert the PUN into the input-output order.
- Execute the order.

The symbolic tape name is either a program parameter compiled with the program, entered as data, or generated. This name must correspond to a unique reel of tape during the running of the program. It is not desirable to specify a specific reel number at compile time; therefore, if the tape name is compiled with the program, this name is usually a functional designation, e.g., an input tape might be designated as INPUT. The specific reel identification is then associated with this symbolic tape name by the use of the TAPES control instruction at job time. For example, let the desired input tape be reel number 1 0234. Then, execution of the following control instruction will relate INPUT to 1-0234:

TAPES 1-0234, INPUT

This method has the advantage that if the input reel number is different on the next execution of the object program, only the TAPES control instruction card need be changed.

#### INTERNAL ASSIGNMENT

The internal method requires that the PUN in an input-output order of a program be defined at compile time. When a program is written, a pre-determined PUN is placed in the input-output order, either directly, or symbolically (if a symbol, it must be defined with an ASGN card). The TAPES control card is used to relate the reel identification with the PUN. As an example, let the PUN chosen be 9, and let the reel identification of the tape used be 1-0234; then, execution of the following control instruction, prior to the running of the object program, will make the proper association.

TAPES      1-0234, 9

#### PUN'S PERMITTED

The PUN's permitted to the user are the numbers 0 through 15. Numbers 0 through 8 refer to specific system tapes, and if used will refer to the system tape designated in the following table:

#### SYS PUN'S

<u>PUN</u>	<u>System Symbolic Tape Name</u>
0	<del>TSYSCR0</del> (scratch tape)
1	TSYSTEM (system tape)
2	TSYSDMP (scratch tape)
3	<del>TSYSCR3</del> (scratch tape)
4	<del>TSYSCR4</del> (scratch tape)
5	TSYSOUT (system output tape)
6	<del>TSYSCR6</del> (scratch tape)
7	TSYSLIB (system library tape)
8	TSYSIN (system input tape)

#### RULE

The following rule must be observed: Any reference (via 32KXORD, or a 32KSYS Tape Handling function) to a PUN greater than 8 must be related to a symbolic tape name by a TAPES control instruction.

Control will be returned to the system via IXCONER if the above rule is violated.

# TAPES

## FUNCTION

Relates a tape name (reel number) written on the job sheet to the tape name used by the program.

## FORMAT

L	Location	Command	Address and Remarks
H		TAPES	<u>par 1 (W), par 2</u>

## PARAMETERS

par 1 - The symbolic tape name (e.g., a reel number or system symbolic tape name) written on the job sheet to identify the tape reel to be mounted by the operator.

(W) (OPTIONAL) - Indicates that par 1 is an output tape and must be write enabled.

par 2 (OPTIONAL) - The symbolic tape name or program unit number by which subsequent programs refer to the tape reel identified by par 1.

H (OPTIONAL) - Indicates that par 1 is to be held for the next job (refer to 1ENDJOB).

## ACTION

The tape tables within 32KSYS are updated to relate par 1 to par 2.

## RESTRICTIONS

1. If par 2 is a program unit number, it must be 9 thru 15 and this must be the first appearance of par 1 within the job.
2. If par 2 is a symbolic tape name, this must be its first appearance within the job.

## REMARKS

1. The TAPES control instruction pertains to the current job only.
2. par 1 is typed on the console typewriter by 32KSYS for tape assignment by the operator (refer to Operating Procedures).
3. The (W) parameter enables 32KSYS to distinguish between legal and illegal write orders to a tape that is not write enabled. However, 32KSYS does not distinguish between legal and illegal write orders to a tape that is write enabled.
4. If par 2 is omitted, subsequent programs may refer to the tape reel by par 1.

**RELEASE****FUNCTION**

Releases a symbolic tape name from its assignment.

**FORMAT**

L	Location	Command	Address and Remarks
		RELEASE	<u>t</u> <sub>1</sub> . . . <u>t</u> <sub>2</sub>

**PARAMETERS**

t<sub>1</sub> . . . t<sub>2</sub> = Symbolic tape names, only.

**ACTION**

Execution of the RELEASE control instruction initiates the following action:

- The tape name(s) specified are released from their specific assignments by removing the name(s) from the tape name list.
- The tape(s) associated with the tape name(s) released may or may not be rewound with lockout. A tape is rewound with lockout if the released tape name was the only reference to the tape.

**REMARKS**

1. Any tape name not in the tape name list will be ignored.
2. System tape names will be ignored.
3. A numeric designation will result in a control line error.



## **SECTION III**

### **SYSTEM TAPES**

Normal 32KSYS use requires nine magnetic tape units which the operator assigns to the following symbolic tape names when needed.  
(Refer to Operating Procedures, Appendix A.)

## **TSYSTEM**

### **TSYSTEM - System Program Tape**

#### **FUNCTION**

**TSYSTEM** is the System Program Tape, containing 32KSYS in image form and System Routines, e.g., TAC, RPLC, TACSERV, COBOL, etc., in RPL form.

**TSYSDMP - System Scratch Tape**

<b>FUNCTION</b>	<b>TSYDMP is a System Scratch Tape used by:</b>
	<ul style="list-style-type: none"><li>● <b>32KSYS as an intermediate tape for dumps and snaps.</b></li><li>● <b>TAC, ALTAC, COBOL, REPORT GENERATOR, and ALTAC3X as a scratch tape.</b></li></ul>
<b>REMARKS</b>	<b>TSYSDMP is rewound when a JOB control instruction is encountered.</b>

## TSYSIN

### TSYSIN - System Input Tape

#### FUNCTION

The System Input Tape contains 32KSYS control instructions in either Code Mode (10 words per card, 12 cards per block) or Image Mode (20 words per card, 6 cards per block).

#### REMARKS

If desired, a special mixed image/code format may be used. A mixed mode System Input Tape is constrained as follows:

- a. The two indicator cards, IMGEIMAGE and CODECODE, specify the mode of the card immediately following the indicator card.
- b. All JOB cards, IMGEIMAGE cards, and CODECODE cards must be on tape in code mode.
- c. The JOB card must contain JOBΔΔΔΔΔΔ in the command field.
- d. No restriction is placed on the positioning within a block of an image mode card except no image mode card may straddle a physical block. Therefore, an IMGEIMAGE card must be used to fill the last 10-word item of a block when the crossover would occur.
- e. The JOB card, in addition to its normal functions, also performs the same function as the CODECODE indicator card. Therefore, cards immediately following the JOB card are assumed to be in code mode until an IMGEIMAGE card is encountered.

**TSYSOUT - System Output Tape****FUNCTION**

TSYSOUT is the System Output Tape on which information is placed for off-line printing and punching. To use the tape, however, the programmer must adhere to the 32KSYS standards concerning the four modes of Data Select described below under FORMAT.

It is used by 32KSYS to print:

- Information to identify jobs.
- All executed control instructions.
- Error information.
- Dumps and Snaps.

It is used by all compilers to:

- Punch binary card images.
- Print edited output.

**FORMAT**

The format of the various types of 32KSYS output, specified by the particular Data Select mode, is described as follows:

**Data Select 0**

Data Select 0 is used to identify blocks to be printed on the High-Speed Printer. The standard print plug-board setting is one-to-one.

**Data Select 1**

Data Select 1 is used to identify blocks for binary card punching in Image Mode (20 words per card, 6 cards per block), with the Control Character SENSE/IGNORE Switch of the Punched Card Controller set to IGNORE when these cards are punched. To aid operators in separating binary decks and associating them with other output from the various jobs, 32KSYS precedes each distinct deck with one card in which all rows of Columns 1 through 8 are punched. For the convenience of operators, any programs using the off-line punch features of TSYSOUT should state via the Console Typewriter the number and type of cards to be punched.

## **TSYSOUT**

### **Data Select 2**

Data Select 2 is used to identify blocks for Hollerith card punching in Code Mode (10 words per card, 12 cards per block), with the Control Character SENSE/IGNORE Switch of the Punched Card Controller set to IGNORE when these cards are punched. As mentioned previously, any program using the off-line punch features of TSYSOUT should state via the Console Type-writer the number and type of cards to be punched.

### **Data Select 3**

Data Select 3 is used to identify blocks containing JOB accounting cards in Code Mode (10 words per card, 12 cards per block). These cards contain the same JOB accounting information as appears in Data Select 0, if an accounting clock is in the system.

### **Data Select 4**

Data Select 4 is used to identify blocks containing data for the X - Y Plotter.

### **WRAPUP**

Whenever the Wrapup function of 32KSYS is used by an operator, the following is written on TSYSOUT.

- End-of-tape sentinels with unconditional stops in each of the following Data Select modes used by 32KSYS:
  1. In Data Select 0, this statement, edited for the High-Speed Printer, appears several times:

THIS IS THE END OF THE TAPE STOP PRINTING

It is followed by a page-eject.
  2. In Data Select 1, a block of filler characters (Octal 32) is edited for punching.
  3. In Data Select 2, ten words are edited to punch a Hollerith-Image card with the first and last rows of all columns punched.
  4. In Data Select 3, a block of filler characters (Octal 32) is edited for punching.

## **TSYSOUT**

**5. In Data Select 4, the following operations for the X - Y Plotter are performed:**

- a. The pen is raised and positioned 0.5 inches toward the top of the paper, and is then positioned at the left margin.**
- b. The pen is lowered and a saw-toothed pattern with a  $45^{\circ}$  diagonal length of 1.12 inches is drawn from left to right.**
- c. The pen is raised and positioned 1.26 inches above the saw-toothed pattern.**
- A sentinel block of Z's is then written, and the tape is rewound with lockout.**

**END OF TAPE**

Whenever end of tape is sensed, the tape assigned to TSYSOUT is rewound with lockout and a new assignment requested for TSYSOUT.

## **TSYSLIB**

### **TSYSLIB - TAC Library Tape**

#### **FUNCTION**

**TSYSLIB** is the normal TAC Library Tape.  
Subroutines should be in relocatable binary format  
because the tape is automatically searched by 32KSYS  
when a REL....SUBS call is encountered.

In general, **TSYSLIB** contains the various macro-  
instructions, generators, and relocatable binary sub-  
routines that are available to the programmer.

**TSYSCR0 - System Scratch Tape**

FUNCTION	TSYSCR0 is used as a scratch tape by TAC, COBOL, REPORT GENERATOR, ALTAC, and ALTAC3X.
REMARKS	TSYSCR0 <u>cannot</u> be assumed to be in a rewound position.

**TSYSCR3**

**TSYSCR3 - System Scratch Tape**

**FUNCTION**            TSYSCR3 is used as a scratch tape by TAC, COBOL,  
                         REPORT GENERATOR, ALTAC and ALTAC3X.

**REMARKS**            TSYSCR3 cannot be assumed to be in a rewound  
                         position.

**TSYSCR4 - System Scratch Tape**

**FUNCTION**

**TSYSCR4 is a System Scratch Tape used by:**

- **TAC as a machine language output tape unless a PROGTAPE control instruction preceded the TAC control instruction.**
- **ALTAC - same as TAC.**
- **COBOL as a positioned scratch tape, (TSYSCR4 is repositioned when the COBOL phase is finished) and as a machine language output tape unless a PROGTAPE control instruction preceded the TAC control instruction.**
- **Report Generator - same as COBOL.**

## **TSYSCR6**

### **TSYSCR6 ~ System Scratch Tape**

#### **FUNCTION**

TSYSCR6 is used as a scratch tape by TAC, COBOL, REPORT GENERATOR, ALTAC, ALTAC3X and ANALYZER.

#### **REMARKS**

TSYSCR6 cannot be assumed to be in a rewound position.

## **SECTION IV**

### **32KSYS XORD**

#### **FUNCTION**

**32KSYS XORD**, the 32KSYS input-output subroutine, provides a simple means of issuing read, write, space, and rewind orders.

#### **FEATURES**

The following are some of the features of XORD:

- The XORD subroutine is contained within memory at all times.
- 32KSYS and the 32KSYS programmer use XORD to execute magnetic tape orders.
- Single and multi-block read, write, and space orders can be issued.
- Rewind and rewind-with-lockout orders can be issued.
- Order checking is automatic. For example, issuing a tape order causes automatic checking of any outstanding order for the unit.
- In many instances, job restarts or cancellations are eliminated. For example, if a unit is in local, XORD allows the operator to put the unit in remote and continue the program.
- In many cases, successive orders for the same tape are executed before tape movement stops.
- Normal use of XORD results in the order being processed by the Input-Output Processor (IOP) when the exit occurs.

## ENTRANCES TO XORD

The subroutine XORD has the following entrances:

- MXORD - an entrance for executing single and multi-block orders, and rewinds.
- XORD - an entrance for executing single-block orders and rewinds.
- CHKORD - an entrance for checking orders.
- SETXORD - an entrance for establishing a new error exit or cancelling a previously established one.
- XTAPE - an entrance for obtaining the logical tape unit number.
- SPORD - an entrance for executing an MXORD order and establishing an error exit effective only for the MXORD order.
- CLORD - an entrance for eliminating an outstanding order, regardless of its status.

## CHECKING ORDERS

For XORD, "checking an order" means "forcing the order to completion without errors." To achieve this, XORD takes corrective action when errors occur and/or calls for operator intervention if necessary. (A Rewind or a Rewind with lockout order is considered complete when accepted.)

There are three methods available to the programmer for checking an order. These methods are listed below according to their probabilities of eliminating start-stop tape time:

- Issue another order to the unit.
- Issue a CHKORD call.
- Set bit 17 of the Input-Output order word.

## EFFICIENT USE OF XORD

Whenever possible, outstanding orders should be checked by issuing another order to the unit, as this minimizes the over-all computer time used by XORD, and offers the greatest probability of eliminating start-stop time. When a single input-output area is used, it is usually

**best to call on MXORD with bit 17 set to zero, continue processing which does not reference the input-output area and then call on CHKORD.**

**Bit 17 set to one should be used only if the programmer would otherwise issue a CHKORD call almost immediately after an XORD or MXORD call.**

## **MXORD**

### **THE MXORD ENTRANCE**

#### **MXORD - Execute Order(s)**

To execute a read, write, space, or rewind order either of the following calls may be used:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
S		MXORD or TMA JMP	<u>ord</u> \$ <u>ord</u> \$ <u>SYS.1MXORD</u> \$

where ord is an input-output order word with the following format:

<u>Bit</u>	<u>Setting</u>	<u>Indication</u>
0	0	Indicates a read or write order.
	1	Indicates a space order. (If bits 40-47 denote rewind, this bit is ignored.)
1-7	0	Not decoded
12-15		The number of blocks (1-15) to be written, read, or spaced.
16	0	Indicates that the order must be accepted before exiting.
	1	Indicates that the order need not be accepted before exiting.
17	0	Indicates that XORD is to exit immediately after the order is accepted.
	1	Indicates that XORD is to exit immediately after the order is complete.
18-23		PUN (Program Unit Number).
24-39		Core Starting Address. This address may be indexed.

<u>Bits</u>	<u>Setting</u>	<u>Indication</u>
40-47		The Input-Output command, which must be one of the following:
	H/91 Read forward	
	H/D1 Read backward	
	H/19 Write	
	H/8A Rewind	
	H/8B Rewind with lockout	
STEPS TAKEN IN ISSUING ORDERS	XORD attempts to issue the order specified in the A Register. If the order is not accepted, XORD proceeds according to the setting of bit 16 of the A Register. If the order is accepted, XORD proceeds according to the setting of bit 17.	
Bit 16	If the order is not accepted, and	
	<ul style="list-style-type: none"> <li>• If bit 16 is one, XORD returns to the calling program without taking further action. (The contents of the A Register will have been made negative.)</li> <li>• If bit 16 is zero, XORD will force acceptance of the order and then proceed according to bit 17 (see Bit 17 below). The A Register will be positive when XORD returns to the calling program.</li> </ul>	
	The following actions are taken, as required, to force acceptance of the order:	
	<ul style="list-style-type: none"> <li>• In the event of a condition which XORD cannot correct (for example, unit in local or missing write-ring) operator intervention is requested by a type-out.</li> <li>• If a previous order to the specified unit is being executed, XORD waits until that order is completed without errors. If XORD detects an error while waiting, appropriate action is taken to correct the error.</li> <li>• If all assemblers are busy because of orders previously issued to other units, XORD takes the appropriate action to free an assembler.</li> </ul>	

## MXORD

### Bit 17

If the order is accepted (see Bit 16), bit 17 is interrogated:

- If bit 17 is zero, XORD returns to the calling program while the order is still being processed by the IOP.
- If bit 17 is one, XORD completes the order before returning to the calling program. Error corrective action is taken if necessary.

The following table shows the effect of various settings of bits 16 and 17:

<u>Bit 16</u>	<u>Bit 17</u>	<u>Conditions on Return to Calling Program</u>
0	0	IOP is processing order. Contents of A Register are positive.
0	1	Order is complete. Contents of A Register are positive.
1	0	If Contents of A Register are positive, Input-Output Processor is processing order.  If contents of A Register are negative, order was not accepted.
1	1	If contents of A Register are positive, order is complete.  If contents of A Register are negative, order was not accepted.

### STATUS OF REGISTERS ON RETURN

When control is returned to the main program from XORD, Registers A, Q, and D have been altered. No index registers are altered, unless a CSA is indexed with a counting index, in which case that index will be incremented once. The A Register will be positive if the order was accepted and negative if not (see Bit 16).

## XORD

### XORD - Executes Order

#### THE XORD ENTRANCE

To execute a rewind or a single-block read, write, or space order either of the following calls may be used:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
S		XORD	<u>ord</u> \$
		or	
		TMA	<u>ord</u> \$
		JMP	SYS.1XORD \$

where ord is an Input-Output order word. Except for the contents of bits 12-15, which must be zeros, the format of this Input-Output order word is the same as that used with the MXORD call.

See MXORD entrance for steps taken in issuing orders.

## CHKORD

### CHKORD - Checks an Order

#### THE CHKORD ENTRANCE

To check an order using the CHKORD entrance either of the following calls may be issued:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
S		CHKORD or TMA JMP	<u>unit \$</u> <u>unit \$</u> <u>SYS.1CHKORD \$</u>

where unit is a word containing the program unit number in bits 18-23.

XORD returns to the calling program when the outstanding order to that unit is completed without errors. If no order is outstanding for that unit, XORD returns immediately.

Index registers are unaffected, the contents of Registers A, Q, and D are altered.

The use of the CHKORD call does not guarantee that the next order for the unit will be accepted. The unit may be in local or the tape may be rewinding as a result of an outstanding Rewind or Rewind with Lockout order.

**CLORD - Terminates an Order****THE CLORD  
ENTRANCE**

To force termination of an outstanding order on a unit, regardless of its status, either of the following calls may be used:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
S		CLORD or TMA	<u>unit</u> \$
		JMP	<u>unit</u> \$ SYS.1CLORD \$

where unit is a word containing the program unit number in bits 18-23.

The CLORD call causes XORD to interrogate the unit and wait until the outstanding order is completed or terminated by errors (XORD does not wait for completion of REWIND orders); a STOP order is then issued for the unit.

A CLORD call should not be issued if the positioning of the tape or the status of the previous order to the unit is important.

**XTAPE - Obtains Logical Tape Unit Number for PUN**

**THE XTAPE  
ENTRANCE**

To obtain the logical tape unit number for the program unit number either of the following calls may be used:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
S		XTAPE or TMA	<u>unit</u> \$ <u>unit</u> \$ SYS.1XTAPE \$
JMP			

where unit is a word which contains, in bits 18-23, the program unit number. When XORD returns control to the calling program, the A Register contains the corresponding tape unit number in bits 18-23. The other bits in the A Register will be the same as the corresponding bits in unit. Registers Q, D, and JA will have been altered on return to the calling program; however, no index registers are altered.

## **ERROR EXITS AND RECOVERY PROCEDURES**

**32KSYS XORD has two basically different error procedures:**

- **A typeout/halt procedure which is used when no error exit has been specified (see below for further details).**
- **An error exit procedure which requires that the user write an error exit subroutine and specify the address of the error exit subroutine by using SETXORD and/or SPORD calls.**

### **ERROR TYPEOUT/HALT**

If no error exit subroutine address has been specified and an error occurs, XORD types an error notice and then halts. Each error notice consists of the symbolic tape name, the logical unit number, and a message indicating the type of trouble. The following illustrates the format of a full error notice:

**TSYSOUT TAPE 5 S1 ERROR**

The following table lists the possible messages and their meanings.

### **ERROR MESSAGES**

<u>Message</u>	<u>Meaning</u>	<u>Suggested Operation Action</u>
IN LOCAL	Unit is mechanically unavailable.	Place tape in remote and press advance.
WRT NABL	Unit should be write enabled.	Write enable the tape and press advance.
ROCKED	XORD has made repeated unsuccessful attempts to fix parity and/or sprocket errors. (five erase resume actions for a write order; ten retries for a	Dump job or press advance to retry.

**ERROR  
MESSAGES  
(Continued)**

<u>Message</u>	<u>Meaning</u>	<u>Suggested Operation Action</u>
	read order, five in each direction).	
ILL.WRT	XORD has been told to write on a tape that is not write-enabled and should not be used as an output tape.	Dump job.
PROG ERR	The calling program has erred (possibly an improper XORD parameter).	Dump job.
SYS ERR	The hardware is malfunctioning (possibly XORD program has been partially clobbered).	Dump job.
DISABLED	Unit became mechanically disabled while executing an order.	Dump job.
BEG TAPE	XORD has been told to process a block prior to first block on tape.	Dump job.
END TAPE	XORD has been told to process a block after last block on tape.	Dump job.
S1 ERROR	A begin block mark has been missed.	Dump job.

**ERROR  
MESSAGES  
(Continued)**

<u>Message</u>	<u>Meaning</u>	<u>Suggested Oper- ation Action</u>
S2 ERROR	An end block mark has been missed.	Dump job.
BAD FIX	A serious error has occurred while XORD was trying to fix parity and/ or sprocket errors.	Dump job.
FAULTY	Some combination of faults has oc- curred or the hard- ware is malfunc- tioning.	Dump job.

## SETXORD

### SETXORD - Establishes Error Exit

#### THE SETXORD ENTRANCE

An error exit may be established by either of the following two calls:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
S		SETXORD or	<u>param \$</u>
	TMA		<u>param \$</u>
	JMP		<del>SYS.</del> ISETXRD \$

where param is a word which has the following form:

P/ERRORX, F40\$

ERRORX is the address of an error exit subroutine and must not be an indexed address. The error exit subroutine thus specified will be used for all orders not issued with SPORD calls until another SETXORD call is given to XORD. A zero error exit address indicates that no error exit subroutine is available (except for orders issued with SPORD calls). At the beginning of each job the operating system issues a SETXORD call specifying a zero error exit address.

**SPORD - Establishes Error Exit for a Particular Order****THE SPORD  
ENTRANCE**

A **special error exit address** may be established (to be used for a particular order) by either of the following two calls:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
S		SPORD or TMA	<u>param \$</u> <u>param \$</u>
		JMP	<u>SYS.1SPORD \$</u>

where param is a word which has the following form:

C/HLTL, CALLWORD;P/SPECIAL.ERRORX, F40 \$

CALLWORD is the address of a parameter suitable for an MXORD call. SPECIAL.ERRORX is the address of a special error exit subroutine which XORD will use if and only if trouble occurs due to the order in location CALLWORD. The special error exit subroutine will not be used for troubles on other units or for errors attributable to the previous/next order to the unit specified in location CALLWORD (the timeout/halt procedure or the error exit subroutine specified by a SETXORD call will be used for these conditions). SPECIAL.ERRORX must not be an indexed address.

## ERROR EXIT

### INTRODUCTION

This discussion is applicable to error conditions other than unit mechanically unavailable and tape not write enabled, which always result in the timeout/halt action (see above). If trouble occurs and an error exit subroutine has been specified (see SETXORD and SPORD calls) XORD saves information required for restarting, which could be clobbered by calls on XORD, and sets a switch to prevent the clobbering of this data in the event of a subsequent error exit. In the following description this switch is referred to as XSWITCH. It is assumed that the error exit subroutine is entered via the following coding

L	Location	Command	Address and Remarks
	RETURNA	JMP	(P) \$
	RETURNB	JMP	(P) \$
	ERRORX	TJM	RETURNA\$
		TJM	RETURNB\$
		INCA	RETURNB\$
			.....

### CONDITIONS ON EXIT

The following information is available to the error exit subroutine when it is called by XORD.

### Q Register

The last order (in MXORD format) issued by XORD to the unit in trouble. Bits 1-6 contain the program unit number and bits 20-23 contain the logical unit number. The core starting address, if applicable, reflects indexing, if any. This order is not involved in the trouble which resulted in XORD calling on the error exit subroutine if the status word indicates no Input-Output Processor faults.

### A Register

An XORD status word for the unit in trouble. Bits are set as described below:

Bits	Meaning
0	Undefined.
1	Bad fix (serious trouble while fix parity/ sprocket error).
2-3	Undefined

## ERROR EXIT

Bits	Meaning
4	Rocked (too many parity/sprocket fix attempts).
5	Program error (possibly an improper XORD parameter).
6	Disabled transport.
7	S1 error.
8	S2 error.
9	Sprocket error.
10	Illegal write order if bit 5 is set otherwise IOP malfunction.
11	Parity error
12	IOP malfunction.
13	Begin tape
14	End tape
15	Space fault
16-19	Undefined
20-23	Number of blocks remaining to be processed as indicated by assembler counters.
24-45	Undefined.
46	Same as bit 1.
47	Same as bit 4

### Indexed Table

A table of connectors and error recovery data used by XORD and available to the error exit subroutine subject to the constraints listed under Error Recovery Rules.

Location	Contents
00,1X	C/JMP, XORD.FINIS; C/JMP, XORD.EXIT\$ where XORD.FINIS is the address of coding which restores main program (program which

## ERROR EXIT

Location	Contents
	called XORD) index registers 1X and 2X and then returns control to the main program. XORD.EXIT is the address of the JMP command which returns control to the main program. (Neither of these JMP commands is to be used without first executing JMP 9, !\$) The contents of Registers A and Q are not altered by use of these JUMP commands. Indicators can be given to the main program via A and Q.
01, 1X	Current callword in MXORD format. The core starting address (if applicable) reflects indexing if any. The program unit number appears in bits 1-6 as well as in bits 18-23.
02, 1X	Main program index registers 1X and 2X. The contents of 1X are in the left address with the C Bit scaled at T16; the contents of 2X are in the right address, with the C Bit scaled at T40.
03, 1X	Undefined.
04, 1X	Input Output Processor order in machine language format which XORD is currently trying to issue. This word is D/0\$ if XORD is currently checking an order CHKORD call or final phase of XORD/MXORD/SPORD call with bit 17 set to zero) rather than trying to issue an order.
05, 1X	Undefined.
07, 1X	Undefined.
08, 1X	Entrance to XORD typeout subroutine. JMPL 08, 1X\$ will result in XORD typeout of an error notice followed by return of control to the command following the JMPL 08, 1X\$ instruction. The contents of 1X will not be altered. See TYPEOUT/HALT for the possible error notices which may occur. If for example the error exit was due to an end of tape error for symbolic tape FILE 04, logical unit number 12, the JMPL 08, 1X\$ the instruction would cause the following typeout:

FILE 04 TAPE 12 BEG TAPE

## ERROR EXIT

Location	Contents
09, 1X	Entrance to an XORD subroutine which resets XSWITCH (see above). If XORD is not restarted via RETURNA or RETURNB, a JMPL 09, 1X must be executed. Control will be returned to the command following the JMPL 09, 1X\$ instruction with the contents of A, C, and all index registers unaltered.
ERROR RECOVERY RULES	The data described above may be used by the error exit subroutine subject to the following rules: <ol style="list-style-type: none"><li>1. No part of the table indicated by the contents of IX may be altered.</li><li>2. A JMPL 9, 1X must be executed if XORD is not restarted via RETURNA or RETURNB.</li><li>3. If corrective action involving calls on XORD itself is taken, a JMPL 09, 1X\$ instruction must not be executed prior to any of these calls.</li><li>4. If XORD or MXORD calls are issued, another error exit may result. Therefore, it is advisable to use SPORD calls in the error exit subroutine, specifying special error exit addresses within the error exit subroutine itself.</li></ol>
ADDITIONAL ERROR RECOVERY INFORMATION	The following information will probably be of use in writing an error exit subroutine. <ol style="list-style-type: none"><li>1. When XORD calls on an error exit subroutine, the unit in trouble is available if bit 6 is zero and the transport is not disabled. XORD has waited for the tape to stop and has issued a STOP order.</li><li>2. When XORD calls on an error exit subroutine, at least one assembler is available.</li><li>3. Unlike previous versions of XORD, an error exit subroutine is not called unless the trouble involves the unit specified by the current call on XORD.</li><li>4. JMP RETURNA will result in the normal XORD TYPEOUT/HALT action. XORD will pretend that it never called the error exit subroutine.</li></ol>

## ERROR EXIT

5. JMP RETURNB\$ will result in restarting XORD as though the trouble and subsequent call on the error exit subroutine had not occurred.

If the contents of the A Register are not zero when JMP RETURNB\$ is executed, XORD will assume that the A Register contains an order which is to replace the order outstanding for the unit in trouble. This order must be in the same format as the contents of the O Register on entry to the error exit subroutine. XORD assumes that this order has been issued by the error exit subroutine and it is used by XORD to update its list of outstanding orders. Use of this feature is not required if the error exit subroutine uses XORD to issue orders.

6. If an error exit occurred because a tape was ROCKED (bit 4 or bit 47 of status word is set to one), the following is applicable:
  - a. For a write order, an erase order has already been issued and completed.
  - b. For a read order, XORD has issued and completed a reverse read order. The block involved in the parity/sprocket trouble will be the next block processed or spaced over, if an order (specifying the original direction) is issued to the unit.
7. When an error exit occurs, the address of the unit in trouble is not necessarily contained in the unit address register of an assembler. This means that while Class A orders and rewind orders will be accepted because an assembler is free and a stop has been issued to the unit in trouble, Class B and C orders may not be accepted.

## SECTION V

### CONTROL LINE FUNCTIONS

The fundamental operations of 32KSYS, such as loading programs into memory, reading and writing on tape, and starting and stopping runs, are called into action by 32KSYS Control Lines - each consisting of a control instruction and its parameters.

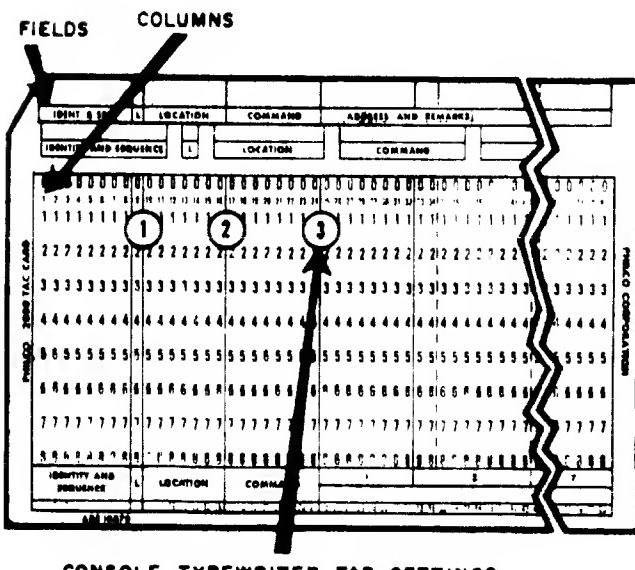
#### SOURCES OF CONTROL LINES

A Control Line must be written in a format similar to that for TAC instructions and may be entered into the computer from either of the following sources:

- Cards via the 32KSYS input tape
- Console Typewriter (manually).

#### CONTROL LINE FORMAT

The format for 32KSYS Control Lines is illustrated below, indicating the location of columns, fields, and Console Typewriter tab settings, including the left margin for the typewriter.



**PHILCO**  
... *A Bell & Howell Company*  
**COMMUNICATIONS AND  
ELECTRONICS DIVISION**

## **PHILCO CODING FORM**

Page..... of .....

CE 1438  
11/64

Information should be written within each field as follows:

- Label Field (Column 9). An L or an R normally placed in this field indicates whether the left or right half of an instruction or an address is to be acted upon by a control instruction.

For the Console Typewriter, the single character is entered before the first tab setting following a carriage return.

- Location Field (Columns 10-16). Normally the octal address of an instruction or data to be acted upon by a control instruction is placed in this field.

For the Console Typewriter, information for this field is to be entered between the first and second tabs following a carriage return.

- Command Field (Columns 17-24). The control line command is to be placed in this field.

For the Console Typewriter, the field is between the second and third tabs following the carriage return.

- Address and Remarks Field (Columns 25-80). The parameters are to be written in this field in the following manner:

1. Parameters must be separated by break characters, i.e., a comma (,), semicolon (;), period (.), or slash (/). No distinction is made between these characters, except where otherwise noted.
2. The last parameter must be followed by a dollar sign if remarks are to follow.

For the Console Typewriter, information for the Address and Remarks field is to be entered following the third tab.

#### REMARKS

1. If any control line command is illegal or, in general, if any of its parameters are illegal, an error type-out will be typed on the Console Typewriter, unless otherwise noted, and control is transferred to IXCONER.

2. If a control line error has occurred while entering control lines from the Console Typewriter, a JOB control instruction must be entered to clear 32KSYS of the error condition.
3. Upon completion of successful action of control instructions, control is generally transferred to the 32KSYS entry 1NXTCON, which requests and attempts to execute the next control instruction.
4. The Philco Coding Form is provided for the user's convenience in writing his programs. Philco 2000 TAC cards are available for punching the information from the Coding Form.

## WRITING CONTROL LINES

As indicated, the command word of 32KSYS control line is to appear in the Command field of a card, and the parameters are to appear in the Address and Remarks field. To simplify explanation of these lines, all capitalized words which appear within the illustration of the instruction format below are to appear on the card exactly as shown. All underlined words in lower case are to be replaced - generally by an L or R in the Label column, an address in the Location field, and parameters selected by the programmer in the Address and Remarks Field.

For example, the instruction LOCTACL is described as:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		LOCTACL	<u>t, id, sent</u>

Although commas are used in format examples throughout the manual, any of the acceptable break characters (, ; . /) are permissible except where otherwise noted.

## CONTROL LINE FUNCTIONS

32KSYS control lines are grouped according to function into nine classifications. These functions, their meaning, and an example of a particular control instruction within each function are listed as follows:

<u>CONTROL LINE</u>	<u>Function</u>	<u>Meaning</u>	<u>Example</u>
<b>FUNCTIONS (Continued)</b>			
	<b>Initialization</b>	<b>Instructions used to prepare the System for compilations or runs.</b>	<b>JOB</b>
	<b>Compilation</b>	<b>Instructions used to call in the compilers and initiate source program conversion from mnemonic coding into machine language.</b>	<b>TAC</b>
	<b>Loading</b>	<b>Instructions used to load an object program into memory and initiate the running of the program.</b>	<b>REL</b>
	<b>Segmentation</b>	<b>Instructions used to divide a program into smaller segments to permit loading and running of only parts of the program in memory at a given time.</b>	<b>SEGMENT</b>
	<b>Tape Handling</b>	<b>Instructions used to read from, write onto, or rewind magnetic tapes.</b>	<b>READF</b>
	<b>Debugging</b>	<b>Instructions used to aid the programmer in diagnosing and patching programs which may be faulty.</b>	<b>DUMP</b>
	<b>Special</b>	<b>Instructions used to perform miscellaneous functions on the computer, such as setting a pseudo toggle register, clearing memory locations, halting an operation, or transferring control to a memory location.</b>	<b>SYSOPS</b>

<u>CONTROL LINE FUNCTIONS (Continued)</u>	<u>Function</u>	<u>Meaning</u>	<u>Example</u>
	Service Routine	Instructions used to call in routines whose functions are tape maintenance, debugging, and program monitoring.	RPLC
	Application	Instructions used to call in application systems which run under control of the 32KSYS monitor.	PERT

## **SECTION VI**

### **INITIALIZATION FUNCTIONS**

The JOB control instruction prepares the System for a new program, and the CONIN instruction specifies the source of control instructions.

## JOB

### JOB

#### FUNCTION

Prepares the System for a new job or series of operations and identifies the job. JOB is the highest level instruction in the 32KSYS hierarchy; therefore each job must begin with a JOB control instruction.

#### FORMAT

L	Location	Command	Address and Remarks
		JOB	<u>ident</u>

#### PARAMETER

ident - 16 or fewer alphanumeric characters \* which identify the job to be processed.

#### ACTION

Execution of the JOB control instruction initiates the following action:

- TSYSDMP is rewound.
- All tapes except those assigned to System names or those assigned to names held for the next job (refer to TAPES Instruction) are rewound with lockout.
- All symbolic tape names except System names and those held for the next job are released.
- An end-of-job message is written on TSYSOUT.
- The RPL, REL, and ABS loaders are reset, i.e., common origin is reset to  $10000_8$ , common storage area reset to 0, etc..
- The DUMP control instruction table and the CONBIT word are cleared.
- The indicator words IIBIT and ISYSOPS are cleared.
- Memory is cleared from location  $10000_8$  to the end of memory.
- The effective end of memory is reset to  $77777_8$ .
- All index registers except 1 and 2 are cleared.
- The System jump vectors are reset.

- The JOB card is typed on the Console Typewriter.
  - The date and time from the Accounting Clock are typed on the Console Typewriter.
  - The edited coding for the JOB page is written on TSYSOUT for printing.
- 
- \* Only those characters which are upper case characters of the Console Typewriter should be used.

**CONIN****CONIN****FUNCTION**

Specifies a new mode of input.

**FORMAT**

L	Location	Command	Address and Remarks
		CONIN	<u>mode</u>

**PARAMETER**

mode - One of the following must be selected:

CODE - Indicates that the next 32KSYS control instruction is in Code Mode.

IMAGE - Indicates that the next 32KSYS control instruction is in Image Mode.

**ACTION**

Whenever CONIN is executed, 32KSYS expects to receive its next control instruction in the mode specified by mode.

## **SECTION VII**

### **COMPILE FUNCTIONS**

This group of control instructions specifies the type of compilation to be made using 32KSYS as an executive routine.

**TAC****TAC - Translator-Assembler-Compiler****FUNCTION**

Initiates a TAC compilation

**FORMAT**

L	Location	Command	Address and Remarks
	format	TAC	<u>source</u> , <u>lib</u> , <u>subs</u> , <u>NOCARDS</u> , <u>NOPROG</u>

**PARAMETERS**format - The format in which the program is to be compiled. If omitted, REL is assumed.

REL - Indicates relocatable binary card format.

ABS - Indicates absolute binary card format.

RPL - Indicates tape (running program language) format.

source - The source of the TAC language program. If omitted, TSYSIN is assumed.

MAGTAPE - Indicates that the source is the system input tape, TSYSIN.

MAGTAPE, t - Indicates that the source is the symbolic tape, t.lib (OPTIONAL) - Indicates that subroutines are to be included in the object program.LIB, t<sub>1</sub> . . . t<sub>n</sub> - Indicates 1-7 symbolic tapes to be used as sources for the subroutines to be copied. If only LIB is specified, the system library tape, TSYSLIB, is assumed.subs (OPTIONAL) - Indicates whether subroutines from the TAC library are to be compiled with the program.

SUBS - Indicates that subroutines are to be compiled.

**NOSUBS** - Indicates that no subroutines are to be compiled.

Note: If omitted, RPL and ABS compilations assume  
SUBS: REL compilations assume NOSUBS.

**NOCARDS** - Indicates that no cards will be produced on TSYSOUT for REL and ABS programs.

**NOPROG** - Indicates that the machine language program will not be produced on the System's or User's Program Tape.

**ACTION**

- The TAC language program is compiled and machine language output, a Code-Edit, and edited cards for punching (if format were ABS or REL) are created.
- The machine language output is produced on the System's or the User's Program Tape unless the NOPROG parameter was specified or compilation errors were detected in the source program. If there were compilation errors, the compiler restores the Program Tape to its position at the start of the compilation. A block of Z's is written on the Program Tape and the tape is positioned immediately prior to that block of Z's. The Code-Edit is prepared and written on TSYSOUT for off-line printing on the High-Speed Printer in Data Select 0.
- The edited cards are produced on TSYSOUT for off-line punching in Data Select 1, unless the NOCARDS parameter was specified.

**REMARKS**

1. If a TAC compilation is successfully completed, the compiler returns to the system via SYS.1NXTCON, where the next control instruction is executed; otherwise it returns via SYS.1COMPER where the remaining control instructions in the JOB are investigated. If the control instruction is one of the following, it is executed; otherwise a search for the next JOB is initiated. This process continues until a JOB instruction is encountered.

## TAC

ALTAC	HLT	OPAL	REWIND
ALTAC3X	IBIT	PRINT	REWINDLO
CLOCK	JOB	RELEASE	SYSOPS
COBOL	LOCTACL	REM	TAC
FORTRAN	NOPRINT	REPORT	TAPES

2. If a serious tape error is detected during a compilation and the operator returns control to the system via memory location 1, the Program Tape is restored to its position at the start of the compilation.
3. The 32KSYS control instruction, SYSOPS, offers the programmer a means of supplying the TAC compiler with optional modes of operation.
4. Edited cards for punching are produced on TSYSOUT even if compilation errors were detected, unless NOCARDS was specified as a parameter.
5. An asterisk in the Address and Remarks field of the TAC control instruction indicates that the TAC language input is on TSYSIN and the library to be used is TSYSLIB. An asterisk is equivalent to MAGTAPE, TSYSIN, LIB, TSYSLIB.
6. A blank Address and Remarks field in the TAC control instruction indicates that the TAC language input is on TSYSIN and there is no library tape. This is equivalent to MAGTAPE, TSYSIN.

CONSOLE  
TYPEWRITER  
NORMAL  
TYPE-OUTS

<u>Message</u>	<u>Explanation</u>
ID IS <u>id</u>	Start of compilation, where <u>id</u> is the program identity.
<u>nnnn</u> CRDS	End of ABS or REL compilation. <u>nnnn</u> is the number of cards produced on TSYSOUT.
<u>nnnn</u> BLKS	End of successful RPL compilation. <u>nnnn</u> is the number of blocks on the Program Tape.

CONSOLE  
TYPEWRITER  
NORMAL  
TYPE-OUTS  
(Continued)

<u>Message</u>	<u>Explanation</u>
ENDCMP	End of successful compilation and the programmer specified NOPROG (in case of RPL) or NOCARDS (in case of ABS or REL). Control is returned to the operating system via SYS.1NXTCON.

CONSOLE  
TYPEWRITER  
ERROR  
TYPE-OUTS

<u>Message</u>	<u>Explanation</u>
TAPE <u>t</u> BAD LIB	Table of contents of library tape is not acceptable. Compiler exits to SYS.1COMPERR.
TAPE <u>t</u> NO LIB	Tape <u>t</u> is not a library tape. Computer halts.
GENERR	Either library or generator error or insufficient parameters in generator call. Generation is terminated and compilation continues.
<u>n</u> DMPTAC	Machine or compiler error. Computer Halts. <u>n</u> indicates the compiler pass where error was detected.
CMPERRS	Source program error detected. The compiler restores the Program Tape to its position at the start of the compilation (before a block of Z's).

# ALTAC

## ALTAC - Algebraic Translator into TAC

### FUNCTION

Initiates an ALTAC III compilation.

### FORMAT

L	Location	Command	Address and Remarks
	<u>format</u>	ALTAC	<u>source</u> , <u>lib</u> , <u>subs</u> , <u>NOCARDS</u> , <u>NOPROG</u> , <u>NOTAC</u>

### PARAMETERS

The parameters for ALTAC are the same as for TAC except for the additional NOTAC parameter.

NOTAC (OPTIONAL) - Indicates the TAC assembly phase of the compilation is to be bypassed. The generated TAC program is on TSYSCR6.

### REMARKS

For a complete description of the parameters, action, and typeouts that can occur during an ALTAC compilation refer to the TAC description.

The following error typeouts can occur during an ALTAC compilation in addition to those described under TAC:

CONSOLE  
TYPEWRITER  
ERROR  
TYPE-OUTS

	<u>Message</u>	<u>Explanation</u>
	EALTAC	A source language error has been detected that would cause generation of an incorrect object program. The TAC assembly is bypassed and control is returned to SYS via SYS.1COMPER.

**ALTAC3X - A Fast Compile-and-Go Version of ALTACIII****FUNCTION**

Initiates an ALTAC3X compilation.

**FORMAT**

L	Location	Command	Address and Remarks
	ABS	ALTAC3X	<u>source</u> , <u>lib</u>

**PARAMETERS**

ABS is the only format in which the object program of an ALTAC3X compilation is produced.

For a complete description of the parameters refer to the TAC description.

**ACTION**

Execution of the ALTAC3X control instruction initiates compilation of the source program, creation of cards, and the following printer output on TSYSOUT:

- The listing of the source program
- The identities of the LIBRARIES USED
- The ASTORS resulting from DIMENSION statements and ASTORS in a TAC insert.
- The PROGRAM MAX indicating the next available memory location following the program
- The address where POOL CONSTANTS begin
- The END CARD JUMP ADDRESS
- A sorted SYMBOL TABLE containing statement numbers, names of variables, and symbols. Except for symbols of the form nnnnnSN (where nnnnn represents a statement number), symbols beginning with a numeric character are not printed.
- The number of ABS cards produced including the dummy PMAX card.

## **ALTAC3X**

- The machine language output is produced on the System's or the user's Program Tape unless compilation errors were detected in the source program. If there were compilation errors, the compiler restores the Program Tape to its position at the start of the compilation.

### **REMARKS**

1. ALTAC3X differs from ALTAC III as follows:
  - a) ALTAC3X does not produce an edited listing of the object program.
  - b) ALTAC3X produces ABS output only, and
  - c) ALTAC3X source programs may not include relocatable binary decks.
2. ALTAC3X comprises the three RPL programs:

PHASE 1: A modified version of the ALTAC III translator,

PHASE 2: A modified version of the first pass of TAC,

PHASE 3: A modified version of the second pass of TAC

3. Remarks 1 and 2 in the TAC description also apply to ALTAC3X.

CONSOLE  
TYPEWRITER  
NORMAL  
TYPE-OUTS

Refer to typeouts and explanations in the TAC description.

CONSOLE  
TYPEWRITER  
ERROR  
TYPE-OUTS

	<u>Message</u>	<u>Explanation</u>
	CMPERRS	Source program error detected. Compilation ends, no object program is produced, and control is returned to the operating system via SYS. !COMPER.
	GENERR	Either library generator error or insufficient parameters in generator call. Generator is terminated and compilation continued.

**CONSOLE  
TYPEWRITER  
ERROR  
TYPE-OUTS  
(Continued)**

<u>Message</u>	<u>Explanation</u>
<b>MCHERR<sub>n</sub></b>	Machine or compiler error. Computer halts with the address field of the HLT instruction representing the location where phase <u>n</u> detected the error.
<b>TAPE <u>n</u> NO LIB</b>	Tape <u>n</u> is not a library tape. Computer halts.
<b>TABOVER</b>	TAC symbol table over-flow. Compilation ends, no object program produced, and control is returned to the operating system via SYS.1COMPER.
<b>TAPE<sub>n</sub> BAD LIB</b>	Table of contents of library tape <u>n</u> is not acceptable. Control is returned to the operating system via SYS.1COMPER.

**PRINTER  
ERROR  
INDICATIONS**

Errors on the Printer output are indicated as follows, depending upon the compiler phase in which the error was detected.

- Source program errors detected during the first phase of the compilation are indicated in the source program listing, and compilation ends after the first phase.
- Errors detected in later phases are indicated in octal in the word ERRINDS. The bits of this word are interpreted as follows:

<u>Bit</u>	<u>Significance</u>
0	LABEL FIELD ERROR
1	COMMAND FIELD ERROR
3	AMBIGUOUS OR CONFLICTING F-BITS
4	LOCATION FIELD ERROR
5	DOUBLE ASSIGNMENT
8	ADDRESS OF NEXT INSTRUCTION CYCLES MEMORY
9	ILLEGAL CONSTANT PREVIOUS CARD
10	ADDRESS FIELD ERROR
11	TOO MANY NAMES, NONAME USED
12	NAME FIELD ERROR, NONAME USED
14	CONTROL CARD ADDRESS FIELD ERROR
18	END CARD ADDRESS ERROR

<u>Bit</u>	<u>Significance</u>
22	ILLEGAL CONTINUATION CARD
23	CSA OF BINARY CARD IMPROPER, USED ZERO
24	BINARY CARD <u>xxxx</u> BAD ( <u>xxxx</u> is an identity and sequence number)
25	DOUBLE ASSIGNMENT WOULD OCCUR IF SYMBOL WERE COMMON
26	SET PARAMETER NOT PREVIOUSLY DEFINED
37	<u>sss</u> NOT DEFINED ( <u>sss</u> represents a symbol)
40	IMPROPER REFERENCE TO INDEX REGISTER

**COBOL****FUNCTION**

Initiates a COBOL compilation.

**FORMAT**

L	Location	Command	Address and Remarks
	<u>format</u>	COBOL	<u>source</u> , <u>lib</u> , <u>sub</u> , NOCARDS, NOPROG, NOTAC

**PARAMETERS**

The parameters are the same as those for the ALTAC control instruction. For a complete description, refer to the parameter explanations of TAC and ALTAC.

**REMARKS**

The COBOL library may be mounted on a tape with the symbolic name COBOLIB instead of being combined with the regular TAC library (TSYSLIB). If COBOLIB is used for the COBOL library, bit 19 of the CONBIT WORD, 1CONBIT, must be set to 1. The setting of this bit is an installation option.

CONSOLE  
TYPEWRITER  
ERROR  
TYPE-OUTS

	<u>Message</u>	<u>Explanation</u>
	NO LIB	A COBOL library required by the source program is not mounted on the correct tape.
	ECOBOL	A source language error has been detected that would cause generation of an incorrect object program. The TAC assembly is bypassed and control is returned to the operating system via SYS.1COMPER.
	MACHERR	A machine or compiler error has occurred.

# OPAL

## OPAL - The Assembly Language for Philco 1000

### FUNCTION

Initiates an OPAL assembly.

### FORMAT

L	Location	Command	Address and Remarks
		OPAL	

### ACTION

Execution of the OPAL control instruction initiates the following action:

- The OPAL language program is assembled and an object program and a code edit are created.
- The format of the object program produced is dependent upon the OPAL "OPTIONS" control card which should immediately follow the I card. ABS (absolute binary) card format is the standard object program format produced by the assembler. The cards are edited for punching with data select 1 and written on TSYSOUT. If the MAGTAPE parameter is specified in the "OPTIONS" control card, the object program is produced in magnetic tape format (1024 characters per block).

In this case the object program is written on TSYSCRO rather than on TSYSOUT. If the n parameter is used, the program is written on tape n instead of on TSYSCRO. (A TAPES control card may be required.) The loader program produced by the assembler, when the MAGTAPE parameter is specified, occupies the first 168 characters of the first block.

- The Code-Edit listing produced by the OPAL assembly is written on TSYSOUT, edited for printing with data select 0.

CONSOLE  
TYPEWRITER  
NORMAL  
TYPE-OUTS

	<u>Message</u>	<u>Explanation</u>
	ID IS <u>id</u>	Program identity, represented by <u>id</u> , is typed out after the first source card is read.

CONSOLE  
TYPEWRITER  
NORMAL  
TYPE-OUTS  
(Continued)

<u>Message</u>	<u>Explanation</u>
PROGRAM TAPE IS <u>n</u>	The Object Program Tape is <u>n</u> . This type-out does not occur unless the program changes normal assignments.
<u>nnn</u> CRDS	This type-out occurs at the end of the assembly for card output; <u>nnn</u> is the number of cards written on the Object Program Tape and/or TSYSOUT.

CONSOLE  
TYPEWRITER  
ERROR  
TYPE-OUTS

<u>Message</u>	<u>Explanation</u>
INSUFFI- CIENT MEMORY	OPAL requires a minimum memory of octal 27000 to run.
OPAL ERROR	Assembler error or machine malfunction.

# REPORT GENERATOR

## REPORT GENERATOR

### FUNCTION

Initiates a REPORT GENERATOR compilation.

### FORMAT

L	Location	Command	Address and Remarks
.	<u>format</u>	REPORT	<u>source</u> , <u>lib</u> , <u>subs</u> , NOCARDS, NOPROG, NOTAC

### REMARKS

For a complete description of the parameters, action, and typeouts that can occur during a REPORT GENERATOR compilation, refer to the TAC and ALTAC descriptions.

The following error typeouts can occur during REPORT GENERATOR compilation in addition to those described under TAC:

CONSOLE  
TYPEWRITER  
ERROR  
TYPE-OUTS

	<u>Message</u>	<u>Explanation</u>
	EREPORT	An excessive number (more than 255) of errors were detected in source program.
	MACHERR	The TAC assembly is bypassed and control is returned to 32KSYS via SYS.1COMPER.
		Computer halts. Generator program flow is in error because of machine failure.

**FORTRAN****FUNCTION**

Initiates a FORTRAN IV compilation.

**FORMAT**

L	Location	Command	Address and Remarks
		FORTRAN	<u>source</u> , <u>lib</u> , NOCARDS, NOPROG, MAP, EDIT, NODEBUG, BATCH

The machine language program output of a FORTRAN IV compilation is in relocatable binary card format written on the system output tape, TSYSOUT, edited for punching, and/or on TSYSCR4 or the User's Program Tape, for subsequent loading and execution.

**PARAMETERS**

source - the source of the FORTRAN IV language program. If omitted, TSYSIN is assumed.

MAGTAPE - Indicates the source is the system input tape, TSYSIN.

MAGTAPE, t - Indicates the source is symbolic tape t in CODE mode (SYSOPS 3 and 4 are not permissible).

lib - Indicates that all called subroutines are to be included in the object program deck.

LIB, t<sub>1</sub>, ..., t<sub>n</sub> - Indicates 1-7 symbolic tapes to be used as sources for the binary relocatable subroutines to be copied.

NOCARDS - Indicates that the object program is not to be written on TSYSOUT edited for punching.

NOPROG - Indicates that the object program is not to be written on TSYSCR4 or the User's Program Tape.

MAP - Indicates that a storage map of the object program is to be written on TSYSOUT, edited for printing.

## FORTRAN

**EDIT** - Indicates that a listing of the object program is to be written on TSYSOUT, edited for printing.

Selection of the EDIT option implies the selection of the MAP option, also.

**NODEBUG** - Indicates that generation of information for the traceback routine is to be inhibited.

**BATCH** - Indicates that multiple compilations are to be performed, the end of which is signaled by the occurrence of a COMPLETE card immediately following the END statement of the last program in the batch. Each compilation in the batch is controlled by the parameters in the FORTRAN call and behaves as any non-batch compilation except that library routines are copied, if the lib parameter was specified, at the completion of the batch rather than at completion of each compilation. Only one copy of each called subroutine is included regardless of the number of compilations calling the subroutine.

### ACTION

Execution of the FORTRAN control instruction causes the following action:

- The FORTRAN IV language program is compiled and machine language output, a source program listing, an object program edit and storage map, and cards edited for punching are produced, according to the options selected.
- The machine language output is produced on the System's or User's Program Tape and the tape is left positioned immediately following the program produced, unless the NOPROG option was selected or compilation errors detected. If compilation errors were detected, the compiler restores the tape to its position at the start of the compilation.
- The source listing, error comments, object program edit, and storage map are produced on the system output tape, TSYSOUT, edited for off-line printing in Data Select 0.
- The object program cards are produced on the system output tape, TSYSOUT, edited for off-line punching in Data Select 1, unless the NOCARDS option was selected or compilation errors detected.

Because some errors are not detected until the last phase of the compilation, the edited cards may be produced even though compilation errors exist. In no case will library routines be included (copied) if errors were detected in the source program.

- If a FORTRAN IV compilation is successfully completed, the compiler returns control to the system via SYS.1NXTCON, where the next control instruction is executed; otherwise it returns via SYS.1 COMPER where the remaining control instructions in the JOB are investigated. If the control instruction is one of the following, it is executed; otherwise a search for the next JOB is initiated. This process continues until a JOB instruction is encountered.

ALTAC	COBOL	RELEASE	REWINDLO
ALTAC 3X	CLOCK	REM	SYSOPTS
IBIT	FORTRAN	REPORT	TAC
LOCTACL	HLT	REWIND	TAPES
NOPRINT	JOB	PRINT	
OPAL			

## REMARKS

1. A serious source program error detected in a batch compilation affects only the output of the compilation of that program containing the error except that the lib option, if selected, is ignored for the entire batch. At the completion of a batch compilation containing at least one serious source program error, the compiler returns control to SYS.1COMPER. See ACTION, above.
2. If the operator terminates the JOB because a serious tape error occurred during compilation, the Program Tape is restored to its position at the start of the compilation.

CONSOLE  
TYPEWRITER  
NORMAL  
TYPE-OUTS

	<u>Message</u>	<u>Explanation</u>
	ID IS <u>id</u>	Start of each compilation, where <u>id</u> is the program identity.
	ENDCOMP	Compilation is complete without serious errors.
	<u>nnnn</u> CRDS	<u>nnnn</u> is the number of cards produced on TSYSOUT.

## FORTRAN

	<u>Message</u>	<u>Explanation</u>
CONSOLE TYPEWRITER ERROR TYPE-OUTS	ID IS NO IDENTITY	First non-blank card did not contain program ID. Compilation continues in normal manner.
	CMPERRS	Compilation is complete. At least one source program error was detected.
	F4DUMP	Machine or compiler error. Computer halts. Press ADVANCE to wrap up compilation.
	<u>id</u> TAPE POSIT. ERR.	Machine or compiler error during the compiler segment identified by <u>id</u> . Computer halts. Press ADVANCE to wrap up compilation.

## SECTION VIII

### LOADING FUNCTIONS

The loading and initiating of program runs is the function of this group of control instructions.

RPL and ABS programs are loaded in a forward sequence from the starting location of the program near the beginning of memory extending toward the end of memory. REL programs are also loaded in a forward sequence, but normally into the end of memory, so that the last word of the program is loaded into the last word of memory. Common storage for REL programs normally begins (as arbitrarily assigned by 32KSYS) at location 1000<sub>8</sub> and may extend to any length required within the space available.

#### ONE WORD SEARCH

A one-word-identity search may be used to locate a program with an identity of more than eight characters. If this method is used, the first word must be terminated by a break character because trailing spaces are significant.

For example, a program with the identity:

PROGA△△1DATA△△△1

appears in RPL format on T4 and has a unique first word of identity. The program may then be loaded by the control instruction:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		RPL	T4, PROGA△△1\$

The same control instruction without the dollar sign

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		RPL	T4, PROGA△△1

results in a transfer of control to 1XCONER. The omission of a break character indicates that the second word of the ID is all spaces and no match can occur.

## GO

### GO Executes Program(s)

FUNCTION                   **Executes programs, loading any programs from TSYSCR4 that were compiled within the current JOB.**

FORMAT	L	Location	Command	Address and Remarks
			GO	<u>t</u>

PARAMETERS               t (OPTIONAL) - The symbolic tape containing binary subroutines.

- ACTION
- If any programs were compiled onto TSYSCR4 within the current JOB, TSYSCR4 is rewound and all programs on TSYSCR4 are loaded.
  - If programs loaded (by previous LOAD instruction or from TSYSCR4) are relocatable then:
    1. Any subroutines required are loaded from t, if t was specified. If t was not specified, then TSYSLIB is searched.
    2. The NAME/SYMBOL list is written on the system output tape, TSYSOUT, edited for printing.

Control is transferred to the program's starting address.

- EXAMPLE ONE
- The following illustrates an RPL compilation with immediate execution:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		JOB	EXAMPLE 1
	RPL	TAC	
		(TAC language deck)	
		GO	

**EXAMPLE TWO**

- The following illustrates multiple relocatable compilations with immediate execution:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	JOB		EXAMPLE 2
	ALTAC		
	(ALTAC		
	language		
	deck)		
	TAC		
	(TAC lan-		
	guage deck)		
	ALTAC		
	(ALTAC		
	language		
	deck)		
	GO		

**EXAMPLE THREE**

- The following illustrates multiple relocatable compilations including the loading of previously compiled program decks, with immediate execution:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	JOB		EXAMPLE 3
	ALTAC		
	(ALTAC		
	language		
	deck)		
	TAC		
	(TAC lan-		
	guage deck)		
	ALTAC		
	(ALTAC		
	language		
	deck)		
	LOAD		
	(binary re-		
	locatable		
	deck)		
	GO		

GO

**EXAMPLE FOUR**

- The following illustrates multiple relocatable compilations including the loading and correcting of previously compiled decks, with immediate execution:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		JOB ALTAC (ALTAC language deck) TAC (TAC lan- guage deck) ALTAC (ALTAC language deck) LOAD (binary re- locatable deck)	EXAMPLE 4
2	25P	ALPHA LOAD (binary re- locatable deck) GO	SAMPLE 01

**LOAD****LOAD - Loads Binary Program Deck(s)****FUNCTION**

Loads binary relocatable or absolute program deck(s) immediately following the LOAD control instruction, up to the next 32KSYS control instruction, for subsequent execution by a GO control instruction.

**FORMAT**

L	Location	Command	Address and Remarks
		LOAD	

**PARAMETERS**

NONE

**ACTION**

If any programs were compiled onto TSYSCR4 within the current JOB, TSYSCR4 is rewound. All program decks immediately following the LOAD control instruction are loaded until a 32KSYS control instruction is encountered.

**REMARKS**

The LOAD instruction must be used in conjunction with the GO instruction when loading relocatable decks because binary subroutines are not loaded and the edited NAME/SYMBOL list is not written on TSYSOUT until a GO is encountered.

**EXAMPLES**

See GO (Examples 3 and 4.)

## **ABS**

### **ABS - ABS Loader Call**

#### **FUNCTION**

Locates and loads an ABS program

#### **FORMAT**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		ABS	<u>t, id, go, i</u>

#### **REMARKS**

The parameters and action are the same as those for the RPL control instruction except for the following:

1. The program designated by id is presumed to be in ABS format.
2. An asterisk, TSYSIN, or an 8 may be used as the first parameter in the ABS control instruction, if a binary absolute deck of the program to be loaded follows the control instruction. Input must be via magnetic tape in IMAGE MODE.

#### **EXAMPLE**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		ABS	*,,GO

Explanation: Execution of this instruction causes the deck of binary cards which follows the instruction on TSYSIN to be loaded into memory. After the program is loaded, control is transferred to the program's starting address. The absence of the second parameter is indicated by a comma.

## REL - REL Loader Call

## FUNCTION

**Locates and loads a REL program.**

## FORMAT

L	Location	Command	Address and Remarks
	<u>m</u>	REL	<u>t<sub>1</sub></u> , <u>id</u> , <u>go</u> , <u>list</u> , <u>subs</u> , <u>t<sub>2</sub></u> ... <u>t<sub>n</sub></u>

## PARAMETERS

m (OPTIONAL) - The loading address of the program.

t<sub>1</sub> - The symbolic tape containing the REL program to be loaded. TSYSIN is excluded if input is in code mode.

An asterisk (\*) is equivalent to TSYSIN.

id - The program identity. If omitted, the first REL program found will be loaded.

The id parameter may be omitted if t<sub>1</sub> is an asterisk (\*), TSYSIN or 8.

go (OPTIONAL) - Indicates immediate or delayed running of a program. If GO is written, control will be transferred to the program's starting address immediately following successful completion of the loading process. If the parameter is omitted, 32KSYS will save the starting address and will execute the next control instruction. A subsequent 32KSYS "JMP\*" control instruction may be used to transfer control to the program's starting address (refer to JMP, Special Functions).

list (OPTIONAL) - This parameter, written as LIST, indicates that all program SYMBOUTS and REFOUTS which have been placed in the NAME/SYMBOL list up to and including this loading are to be edited for printing and written on TSYSOUT immediately, before transferring control to the program ("GO" or "JMP"). If the specified program is not loaded successfully, the NAME/SYMBOL list, if any, will be processed as above, regardless of whether or not LIST was specified.

## REL

subs (OPTIONAL) - This parameter, written as SUBS, indicates that the library tape(s) will be searched for required subroutines to be loaded if there are any undefined symbols remaining at the completion of the loading process.

SUBS, t<sub>2</sub>, ..., t<sub>n</sub> - 1-7 symbolic tapes which designate the libraries containing binary subroutines that may be used by the program just loaded.

Tapes are searched in the order listed in the parameters.

### ACTION

The action is the same as that for the RPL control instruction except for the following:

- The program id is presumed to be REL format.
- If undefined symbols still exist following loading of the designated subroutines from the library tape(s) and GO is specified, the NAME/SYMBOL list will be edited and placed on TSYSOUT for printing. Control will then be transferred to IXCONER.

Note: Binary library tapes must contain a sentinel block of W/ZZZZZZZZ following the last subroutine.

- If there are undefined symbols, the SUBS parameter is not specified, and GO is specified, the same error procedure as above will occur.
- If a symbol is defined more than once, it will not be considered an error unless the symbol is referenced.
- The unique symbol LOADSETS.AVAILMEM has a special definition. If it appears first as a referenced symbol, it is immediately defined as being the location one less than the origin of the program just loaded. This origin is then reduced by one to preserve the location. If this symbol appears first as a defined symbol, that definition is given to it. (Refer to special function PORIG.)
- If location LOADSETS.AVAILMEM appears on the NAME/SYMBOL list, its left and right address portions, respectively, will contain, after loading, the

sum of Common origin and the size of Common, and the address of the smallest program origin greater than zero. (Refer to SEGMENT and MASTER, Segmentation.)

- If the LIST parameter is specified, the NAME/SYMBOL list will be edited and placed on TSYSOUT for off-line printing.
- REMARKS**
1. Before control is given to the loaded REL program(s) the NAME/SYMBOL list is cleared.
  2. REL programs are loaded in a forward sequence, normally into the end of memory or contiguous with the last REL program loaded. However, if a loading address is specified, the REL program is loaded forward beginning at m.
  3. m must be greater than 100018.
  4. If the source (t) of a REL program is specified as TSYSIN or \*, the program must immediately follow the REL control instruction and must be in image mode.

**EXAMPLE ONE**

This example illustrates the loading of a single REL program.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		REL	4, TRAJ, GO, LIST, SUBS, LIBC, LIBA

**Explanation:** Execution of this instruction causes the REL program TRAJ, to be loaded into the end of memory from the tape assigned to the system program unit number 4.

Subroutines required to satisfy undefined program symbols are to be loaded first from LIBC, then LIBA. TSYSLIB will not be searched, since it was not specified.

## **REL**

### **EXAMPLE TWO**

This example illustrates the joint loading of two REL programs.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		REL	4, DIFF1\$
		REL	4, DIFF2, GO, LIST, SUBS

**Explanation:** Execution of these instructions causes the REL program, DIFF1 to be loaded from the tape assigned to the system program unit 4, into the end of memory and a second REL program, DIFF2, to be loaded into memory adjoining DIFF1. Subroutines required by either program to satisfy undefined symbols are to be loaded from TSYSLIB. When both programs and all subroutines are loaded, the NAME/SYMBOL list is edited and placed on TSYSOUT for off-line printing. Control is then transferred to the starting address of DIFF2.

**Note:** For making corrections to relocatable binary programs, refer to REMARK 2 of OCT, Debugging Functions.

### **EXAMPLE THREE**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		REL	TSYSCR4, JOE,,,SUBS
		PORIG	JOE
245P		ALPHA	PROGRAMA
		JMP	*

**Explanation:** Execution of these instructions causes program JOE to be loaded from the tape assigned to TSYSCR4 with subroutines. However, since the ALPHA correction is to be made to JOE and the P-counter is set to the last program loaded (a subroutine) the command PORIG (Special Functions) must be given before the ALPHA command.

**RPL - RPL Loader Call****FUNCTION**

Locates and loads an RPL program.

**FORMAT**

L	Location	Command	Address and Remarks
		RPL	<u>t</u> , <u>id</u> , <u>go</u> , <u>i</u>

**PARAMETERS**

t - The symbolic tape containing the RPL program to be loaded.

id - The program identity. If omitted, the first RPL found will be loaded.

go (OPTIONAL) - Indicates immediate or delayed running of program. If GO is written, control is to be transferred to the program's starting address immediately following successful completion of the loading process. If the parameter is omitted, 32KSYS saves the starting address and executes the next control instruction. A subsequent 32KSYS "JMP" instruction may be used to transfer control to the program's starting address. (Refer to JMP, Special Functions.)

i (OPTIONAL) - An octal number to be added to the normal loading addresses of the program being loaded.

**ACTION**

- t is searched forward for program id until id is found either as an RPL identifier, or as one of the ID's included in an index block, or, until a sentinel block of Z's is found. If id is found in an index block, or the sentinel block is encountered, the tape is searched backward. If id is still not found, control will be transferred to IXCONER.
- When the designated program is located, it is loaded into memory.
- If GO is not specified, control will be transferred to INXTCON.
- If GO is specified, control will be transferred to the program's starting address.

**REMARKS**

1. If t is specified as TSYSIN, exit will be made to IXCONER.

## RPL

2. If an effective loading address of a program, after modification by i is less than 10000<sub>8</sub>, exit will be made to 1XCONER.

### EXAMPLE ONE

This example illustrates the use of an RPL call with the program to be executed immediately.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		RPL	T4, SOLOMON23, GO

Explanation: Execution of this instruction causes RPL program SOLOMON23 to be loaded from tape T4 and control to be transferred immediately to its starting address.

### EXAMPLE TWO

This example illustrates the use of an RPL call with delayed execution of the program.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		RPL	4, SOLOMON23
		.	(Other SYS control instructions)
		JMP	*

Explanation: Execution of these instructions causes RPL program SOLOMON23 to be loaded from the tape assigned to the system program unit number 4. When the 32KSYS "JMP \*" control instruction is encountered, control will be transferred to the program's starting address.

## SECTION IX

### SEGMENTATION FUNCTIONS

Two 32KSYS control instructions, SEGMENT and MASTER provide the facilities for dividing a relocatable binary program, too large to be run with available memory, into smaller segments. Each of these segments may then be contained individually in the memory area.

#### SEGMENTATION PROCESS

The segmentation process generates segments by writing one or more program components (which are loaded as a single unit into memory) onto magnetic tape in RPL forma

The segmentation process takes place in three phases: first, the compilation of components as individual smaller programs in REL format; second, the generation of segments in RPL format; and third, the loading and running of the segmented program in RPL format. The 32KSYS control instructions SEGMENT and MASTER are used in the second phase.

When segments are used in a running program, they may be independent of one another, or they may contain cross-references. If they are independent, they may be generated one at a time during the second phase of the segmentation process. If any contain cross-references, the referencing must be done either via the Common area of memory or via a Master segment which is contained in memory during the generation of all segments.

The SEGMENT instruction generates segments by writing the components just loaded onto magnetic tape. The MASTER control instruction is used to indicate that the component(s) just loaded are to form a Master segment, and are to remain in memory during the generation of succeeding segments.

#### LOADING SEGMENTED PROGRAMS

Program segments may be loaded by the 32KSYS internal loader, which may be called by either of the following methods:

- Via the 32KSYS Generator LOADGEN (Library Routines), which will create the necessary TAC language coding.
- Via the 32KSYS entry l INTLD (32KSYS Entries), which provides a transfer of control to the internal loader during the running of the program.

**SEGMENT - Defines a Program Segment****FUNCTION**

Causes program components loaded into memory in REL format to be written as a segment onto magnetic tape in RPL format.

**FORMAT**

L	Location	Command	Address and Remarks
		SEGMENT	<u>t</u> , <u>id</u> , <u>master</u>

**PARAMETERS**

- t - The symbolic tape onto which the segment is to be written. If TSYSTEM or TSYSOUT is designated, exit will be made to IXCONER.
- id - 16 or fewer alphanumeric characters which will be the identity of this particular segment. Spaces are significant.
- master (OPTIONAL) - A parameter written as MASTER which enables a Master segment to be written on tape.

**ACTION**

Execution of a SEGMENT control instruction initiates the following actions:

- Examines the NAME/SYMBOL list associated with the particular segment of the program to determine if there are any undefined symbols. Undefined symbols cause exit to IXCONER.
- Writes the portion of memory between the most recent program origin (set by REL loader) and the beginning of the Master Segment, if a Master segment has been defined, onto t in RPL format. If a Master segment has not been defined, that portion of memory between the most recent program (set by REL loader) and the end of memory will be written onto t.
- Writes a sentinel block of Z's onto t following the segment, and positions the tape at the beginning of this block. If another segment is to follow, it will be written over the sentinel block and a new sentinel block will be written following the new segment. This procedure is repeated for all segments, including the Master segment, thus insuring the presence of a sentinel block after the final segment only.

**SEGMENT**

- Clears to fixed-point zeros that portion of memory occupied by the segment after the segment is written onto t.
- Clears the NAME/SYMBOL list of all names and symbols defined by the particular segment, excluding the Master segment list, if used.
- Writes onto t in FPL format that area between the beginning of one Master segment and the beginning of the next (or the end of memory, if only one master segment is in memory), if the master parameter is used. In addition, it clears the memory area and the NAME/SYMBOL list used by that Master Segment.

**REMARKS**

Symbolic Location LOADSETS.AVAILMEM may be defined in every segment if it is not defined in any associated Master segment.

**EXAMPLE ONE**

This example illustrates the use of SEGMENT without a master parameter. Assume that the first segment (SEG 1) of a large program consists of components PROG 1 and PROG 2, and that these components are in REL format on the tape assigned to system program unit number 4.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		REWIND	6
		REL	4, PROG 1
		REL	4, PROG 2,, LIST, SUBS
		SEGMENT	6, SEG 1

Explanation: Execution of these instructions causes the two components to be loaded and written in RPL format onto the tape assigned to system program unit number 6.

## SEGMENT

**EXAMPLE TWO**

This example illustrates the use of SEGMENT with the master parameter. Assume that EXEC, SUBEXEC1, and SUBEXEC2 are Master segments. APROG, BPROC, CPROG, and DPROG each constitute segments.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		REWIND	6
		REL	4, EXEC, , LIST, SUBS
		MASTER	
		REL	4, SUBEXEC1, , LIST, SUBS
		MASTER	
		REL	4, APROG, , LIST, SUBS
		SEGMENT	6, SEG1A
		REL	4, BPROC, , LIST, SUBS
		SEGMENT	6, SEG1B
		SEGMENT	6, SEG1, MASTER
		REL	4, SUBEXEC2, , LIST, SUBS
		MASTER	
		REL	4, CPROG, , LIST, SUBS
		SEGMENT	6, SEG2A
		REL	4, DPROG, , LIST, SUBS
		SEGMENT	6, SEG2B
		SEGMENT	6, SEG2, MASTER
		SEGMENT	6, EXEC, MASTER

Explanation: Execution of these instructions causes the six segments to be written in RPL format onto the tape assigned to system program unit number 6. The RPL 6, EXEC, GO causes the overall program to be run.

## MASTER

## MASTER - Defines a Master Segment

**FUNCTION**      Indicates to 32KSYS that (1) a segmenting process using a Master segment is about to take place, and (2) a Master segment consisting of one or more REL components has been loaded and is to remain in memory during the forthcoming loading and writing of the various segments. The Master segment remains in memory during segmentation so that symbolic cross-references may be made with succeeding segments associated with it.

FORMAT	L	Location	Command	Address and Remarks
			MASTER	

**PARAMETERS**      None

**ACTION**      Stores the Master segment's starting address in a list. This address is taken from the last relocatable END card of a Master component to be loaded that contains a starting address. If no component contains a starting address, zero will be stored.

- REMARKS**
1. If a MASTER control instruction is given, control may be transferred to a Master segment still in memory by the use of a "JMP\*" control instruction, if the Master segment has not been written.
  2. Each additional MASTER control instruction forces information about the preceding Master segment to be stored in a push-down list in memory. This information is restored when the succeeding Master segment is written out.
  3. Refer to REMARKS of SEGMENT.

**EXAMPLE**      Assume that the Master segment consists of two components in relocatable binary format with ID's of EXEC1 and EXEC2, both contained on the tape assigned to system program unit number 4.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	JOB	XRAY	
	REL	4, EXEC1	
	REL	4, EXEC2,, LIST, SUBS	
	MASTER		

**Explanation:** Execution of these instructions causes the two components to be loaded into memory, forming a Master segment and retaining the segment in memory until all other segments are loaded and written onto tape. The starting address of the Master segment is one of the following:

- The starting address of EXEC1, if EXEC2 has no starting address.
- The starting address of EXEC2, if it contains a starting address. EXEC2 takes precedence because it is the last to be loaded.
- Zeros, if neither EXEC1 nor EXEC2 contain a starting address.

**Note:** TAC library subroutines never contain starting addresses.

## **SECTION X**

### **TAPE HANDLING FUNCTIONS**

This group of control instructions is concerned specifically with magnetic tape operations. These operations include rewinding tapes, reading tapes forward and backward, writing on tapes, locating on tapes, writing sentinels, and locating sentinels. 32KSYS XORD is used for all tape movements by these control instructions.

## REWIND

### REWIND - Rewinds Magnetic Tape

FUNCTION            Rewinds magnetic tapes.

FORMAT

L	Location	Command	Address and Remarks
		REWIND	<u>t</u> , . . . , <u>t</u>

PARAMETERS        t - The symbolic tape to be rewound.

ACTION            t, . . . , t are rewound.

## REWINDLO - Rewinds Magnetic Tape with Lockout

## FUNCTION

Rewinds magnetic tapes with lockout.

## FORMAT

L	Location	Command	Address and Remarks
		REWINDLO	<u>t</u> , ..., <u>t</u>

## PARAMETERS

t - The symbolic tape to be rewound with lockout.

## REMARKS

Operators are instructed that any tape rewound with lockout is subject to removal.

## ACTION

- Program unit numbers and all names associated with t, ..., t are released from their specific assignments by removing the names from the tape name list.

## READF

READF - Reads Magnetic Tape Forward

### FUNCTION

Spaces forward and/or reads blocks forward from a specified magnetic tape into memory.

### FORMAT

L	Location	Command	Address and Remarks
		READF	<u>t</u> , <u>nbs</u> , <u>nbp</u> , <u>addr</u>

### PARAMETERS

t - The symbolic tape to be spaced and/or read forward.

nbs (OPTIONAL) - The number (decimal) of blocks to be spaced forward. If nbp below is omitted, nbs must be specified.

nbp (OPTIONAL) - The number (decimal) of blocks to be read forward. If nbs is omitted nbp must be specified.

addr (OPTIONAL) - The address into which the data on the tape is to be read. If nbp is omitted or is zero, addr may be omitted.

### ACTION

t is spaced forward nbs number of blocks, and nbp number of blocks are read into memory beginning at addr.

### EXAMPLE ONE

This example illustrates the use of a space-read command.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		READF	6, 2, 5, 15000

Explanation: Execution of this instruction causes the tape assigned to system program unit number 6 to be spaced forward two blocks, and the next five blocks to be read into memory starting at location 15000<sub>8</sub>.

### EXAMPLE TWO

This example illustrates the use of a read-only command.

**READF**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		READF	T-024, , 36, 25000

**Explanation:** Execution of this instruction causes 36 blocks of T-024 to be read into memory starting at location 25000<sub>8</sub>. No spacing of blocks is to take place prior to the read.

**EXAMPLE THREE**

This example illustrates the use of a space-only command.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		READF	T-023, 160

**Explanation:** Execution of this instruction causes T-023 to be spaced forward 160 blocks. No reading is to take place.

## **READB**

### **READB - Reads Magnetic Tape Backward**

#### **FUNCTION**

Space backward and/or reads blocks backward from a specified magnetic tape into memory.

#### **FORMAT**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		READB	<u>t</u> , <u>nbs</u> , <u>nbp</u> , <u>addr</u>

#### **REMARKS**

The parameters, action, and remarks concerning READB are the same as those for the READF instruction except that spacing and/or reading is performed backward on the tape.

#### **EXAMPLE**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		READB	TSYSCR0, 5, 10, 32000

Explanation: Execution of this instruction causes TSYSCR0 to be spaced backward 5 blocks, and the next 10 blocks to be read backward into memory starting at location 32000<sub>8</sub>.

**WRITE - Writes on Magnetic Tape****FUNCTION**

Spaces forward and/or writes a specified number of blocks from memory onto an output magnetic tape.

**FORMAT**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		WRITE	<u>t</u> , <u>nbs</u> , <u>nbp</u> , <u>addr</u>

**PARAMETERS**

t - The symbolic tape to be spaced forward and/or written upon.

nbs (OPTIONAL) - The number (decimal) of blocks to be spaced forward. If nbp below is omitted nbs must be specified.

nbp (OPTIONAL) - The number (decimal) of blocks to be written. If nbs is omitted, nbp must be specified.

addr (OPTIONAL) - The address from which data is to be written. If nbp is omitted or is zero, addr may be omitted.

**REMARKS**

The remarks and action for the WRITE control instruction are the same as those for the READF instruction except that writing onto tape rather than reading from tape is performed.

**EXAMPLE**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
----------	-----------------	----------------	----------------------------

WRITE	3, 3, 6, 17500
-------	----------------

Explanation: Execution of this instruction causes the tape assigned to system program unit number 3 to be spaced forward three blocks, and the next six blocks to be written starting from location 17500<sub>8</sub>.

## LOCSENT

### LOCSENT - Locates a Sentinel on Magnetic Tape

#### FUNCTION

Searches forward on a tape for a sentinel block which contains a designated sentinel as each of the first 120 words.

#### FORMAT

L	Location	Command	Address and Remarks
		LOCSENT	<u>t, sent</u>

#### PARAMETERS

t - The symbolic tape to be searched.

sent (OPTIONAL) - The sentinel word. If omitted, Z's are assumed.

#### ACTION

- t is searched forward for a sentinel block with sent as each of its first 120 words.
- When the designated sentinel block is found, t is positioned immediately past the sentinel block.
- If the sentinel is not found, the computer will encounter a serious tape error.

#### EXAMPLE

L	Location	Command	Address and Remarks
		LOCSENT	6, ZZZZZZZZ

Explanation: Execution of this instruction causes the tape assigned to program unit number 6 to be searched forward for a sentinel block which contains all Z's in the first 120 words. After the sentinel block is located, the tape is positioned immediately past the block.

**LOCABS, LOCRPL,  
LOCTACL,LOCREL**

**LOCABS, LOCRPL, LOCTACL, LOCREL - Locates  
a Program on Magnetic Tape**

**FUNCTION**

Searches a tape for specified TAC language, RPL, ABS or REL programs.

**FORMAT**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		LOCABS	<u>t, id, sent</u>
		LOCTACL	<u>t, id, sent</u>
		LOCREL	<u>t, id, sent</u>
		LOCRPL	<u>t, id, sent</u>

**PARAMETERS**

t - The symbolic tape to be searched.

id - The program to be located.

sent (OPTIONAL) - The sentinel word. If omitted, Z's are assumed.

**ACTION**

- t is searched forward for program id until either id is located or a sentinel block of sent is encountered. If the sentinel block is encountered prior to the id, the tape will be searched backward. If the id is still not found, exit will be made to 1XCONER. (If LOCRPL, refer also to RPL, Loading Functions).
- After the designated program is located, the tape is positioned immediately prior to the first block of the program.

**EXAMPLE ONE**

Assume that Tape T-024 contains the following TACL programs:

PART 1  
PART 2 040162  
PART 3 041062

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		LOCTACL	T-024, PART 3 041062
		LOCTACL	T-024, PART 3\$

**LOCABS, LOCRPL,  
LOCTACL, LOCREL**

Explanation: Execution of either of these instructions causes the third program on the tape to be located.

**EXAMPLE TWO**

Assume that Tape ALPHA contains the following RPL programs:

PART 1  
PART 2 040162  
PART 3 041062

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		LOCRPL	ALPHA, PART 3 041062
		LOCRPL	ALPHA, PART 3\$

Explanation: Execution of either of these instructions causes the third program on the tape to be located.

## WRTABS

### WRTABS - Writes on Magnetic Tape in ABS Format

#### FUNCTION

Writes words between designated memory locations onto magnetic tape in absolute binary card (ABS) format and assigns a program identity to the data written as the new ABS program.

L	Location	Command	Address and Remarks
<u>z</u>		WRTABS	<u>t</u> , <u>id</u> , <u>addr</u> <sub>1</sub> , <u>addr</u> <sub>2</sub> , <u>addr</u> <sub>3</sub> , *

#### PARAMETERS

z (OPTIONAL) - The character Z- which indicates that every word (including fixed point zeros) in the program will be written onto t. If omitted, leading words of zeros will be omitted from each absolute binary card produced. If trailing words of zeros are present, they will be included to complete a card.

t - The symbolic tape on which the data in ABS card format is to be written.

id - The identity of the new ABS program. The first four characters of id are used to provide an ID in Hollerith code for Columns 1-4 of the ABS card output.

addr<sub>1</sub> (OPTIONAL) - The first address from which information is to be written on tape. If omitted, addr<sub>1</sub> will be program origin (set only by REL loader).

addr<sub>2</sub> (OPTIONAL) - The last address from which information is to be written on tape. If omitted, addr<sub>2</sub> will be effective end of memory +1.

addr<sub>3</sub> (OPTIONAL) - The starting address of the new ABS program. An asterisk (\*) indicates that the address stored in 1NTRYJA will be used. (1NTRYJA normally contains the address of a location in the current object program to which control is to be transferred from 32KSYS). If omitted, addr<sub>1</sub> will be used.

## WRTABS

i (OPTIONAL) - An octal number to be added to addr <sub>1</sub> when the loading address of the new ABS program is formed. The number is also added to addr <sub>3</sub> to form the program starting address. The addition of the values takes place modulo machine size.

### ACTION

- All information between addr <sub>1</sub> and addr <sub>2</sub> will be written onto t in absolute binary card (ABS) format. If Z is written as the z parameter, words of zeros will also be included in the output.
- Two sentinel blocks of Z's are written on the output tape following the new ABS program.
- t is positioned immediately prior to the first block of Z's.
- The message: id ORIGIN addr <sub>1</sub> n CARDS will be written on TSYSOUT.
- ID is id n CARDS is typed if the ABS program is written on TSYSOUT.

### EXAMPLE ONE

This example illustrates the use of WRTABS without an offset address.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		WRTABS	T-024, JACKSON, 20000, 30000\$

Explanation: Execution of this instruction causes the contents of all memory locations from 20000<sub>8</sub> to 30000<sub>8</sub>, except words of zeros, to be written in absolute binary format onto tape T-024. The program identity of the new ABS program is JACKSON.

The address specified by addr <sub>1</sub>, 20000<sub>8</sub>, is to be used as the starting address for the ABS program, as addr <sub>3</sub> is omitted. Two sentinel blocks of Z's are written following the new ABS program, and the tape is positioned immediately prior to the first block of Z's.

### EXAMPLE TWO

This example illustrates the use of WRTABS with an offset address.

## WRTABS

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
Z		WRTABS	ALPHA, ABLE, 20000 25000, 22000, 600

Explanation: Execution of this instruction causes the contents of all memory locations from 20000<sub>g</sub> to 25000<sub>g</sub>, including words of zeros, to be written in absolute binary format onto tape ALPHA. The program identity of the new ABS program is ABLE. The location into which ABLE is to be loaded is 20600<sub>g</sub>. The starting address of the new ABS program is to be 22600<sub>g</sub>. Two sentinel blocks of Z's are written following the new ABS program, and the tape is positioned immediately prior to the first block of Z's.

### EXAMPLE THREE

This example illustrates the use of WRTABS with TSYSOUT(5) as the output tape.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
Z		WRTABS	5, GOOD CHEER, 16000, 22000

Explanation: Execution of this instruction causes the contents of all memory locations from 16000<sub>g</sub> to 22000<sub>g</sub>, including words of zeros, to be written in absolute binary format onto TSYSOUT. The program identity of the new ABS program is GOOD CHEER. The address specified by addr<sub>1</sub>, 16000<sub>g</sub>, is to be used as the starting address for the ABS program, as addr<sub>3</sub> is omitted. WRTABS transfers binary card images to this tape for off-line punching in Data Select 1, Image Mode (20 words per card, six cards per block).

## WRTRPL

### WRTRPL - Writes on Magnetic Tape in RPL Format

#### FUNCTION

Writes words between designated memory locations onto magnetic tape in RPL format and assigns a program identity to the data written as a new RPL.

#### FORMAT

L	Location	Command	Address and Remarks
		WRTRPL	<u>t</u> , <u>id</u> , <u>addr</u> <sub>1</sub> , <u>addr</u> <sub>2</sub> , <u>addr</u> <sub>3</sub> , <u>i</u>

#### PARAMETERS

t - The symbolic tape excluding TSYSOUT, on which the RPL is to be written.

id - The identity of the new RPL.

addr<sub>1</sub> (OPTIONAL) - The first address from which information is to be written on tape. If omitted, addr<sub>1</sub> will be programmed origin (set only by REL loader).

addr<sub>2</sub> (OPTIONAL) - An address from which information is to be written on tape. If omitted, addr<sub>2</sub> will be the effective end of memory + 1.

addr<sub>3</sub> (OPTIONAL) - The starting address of the RPL program. An asterisk (\*) indicates that the address stored in LNTRYJA will be used. (LNTRYJA normally contains the address of a location in the current object program to which control is to be transferred from 32KSYS). If omitted, addr<sub>1</sub> will be used.

i (OPTIONAL) - An octal number to be added to addr<sub>1</sub> when the loading address of the new RPL program is formed. The number is also added to addr<sub>3</sub> to form the program starting address. The addition of the value takes place modulo machine size.

#### ACTION

- All information between memory locations addr<sub>1</sub> and addr<sub>2</sub> will be written on t in RPL with a program identity of id.
- Two sentinel blocks of Z's are written on Tape t.

- t is positioned immediately prior to the first block of Z's.
- The message: RPL ORIGIN addr  
id RPL n BLOCKS  
 will be written on TSYSOUT.

**EXAMPLE ONE**

This example illustrates the use of WRTRPL without an offset address, following a REL load, and using symbolic addresses for addr<sub>1</sub> and addr<sub>2</sub>.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	2000	REL WRTRPL	4, DIFF ALPHA, GOODBYE, DIFF.START, DIFF.END

Explanation: Execution of the REL instruction cause the REL program DIFF to be loaded into memory starting at location 20000<sub>8</sub> from the tape assigned to system program unit 4. Execution of the WRTRPL instruction causes the contents of memory locations from DIFF.START to DIFF.END to be written in RPL format onto Tape ALPHA with an ID of GOODBYE. DIFF.START and DIFF.END are defined by the REL loader in the NAME/SYMBOL list if TAC SYMBOUT instructions at compile time, or a 32KSYS SYMDEF instruction at run time, permit recognition of the symbols. If the symbols are not defined, control will be transferred via 1XCONER. DIFF.START will be used as the starting address for the RPL since addr<sub>3</sub> is omitted. Two sentinel blocks of Z's are written following the new RPL on Tape ALPHA then ALPHA is repositioned immediately prior to the first block of Z's.

**EXAMPLE TWO**

This example illustrates the use of WRTRPL with an offset address.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		WRTRPL	T-023, HELLO, 20000, 25000, 22000, 325\$

Explanation: Execution of this instruction causes the contents of memory locations from 20000<sub>8</sub> to 25000<sub>8</sub>

## WRTRPL

to be written in RPL format onto Tape T-023. The program identity of the new RPL is HELLO. The location into which HELLO is to be loaded is  $20325_8$ . The starting address of the new RPL is to be  $22325_8$ . Two sentinel blocks of Z's will be written following the new RPL; then the tape will be positioned immediately prior to the first block of Z's.

**WRTSENT - Writes a Sentinel on Tape****FUNCTION**

Writes a sentinel block of 128 words of a designated configuration onto a specified magnetic tape.

**FORMAT**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		WRTSENT	<u>t, sent</u>

**PARAMETERS**

t - The symbolic tape to be written upon.

sent - The sentinel word to be used as each of the 128 words of the sentinel block.

**ACTION**

A sentinel block of 128 words of sent is written onto t. t remains positioned following the sentinel.

**EXAMPLE ONE**

This example illustrates the use of a WRTSENT command with an eight-character sentinel.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		WRTSENT	6, ZZZZZZZZ

Explanation: Execution of this instruction causes a sentinel block containing 128 words of Z's to be written onto the tape assigned to system program unit number 6.

**EXAMPLE TWO**

This example illustrates the use of a WRTSENT command with a sentinel of fewer than eight characters.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		WRTSENT	NINE, XXX

Explanation: Execution of this instruction causes a sentinel block containing 128 words of 00000XXX to be written onto symbolic tape NINE.

## SECTION XI

### DEBUGGING FUNCTIONS

Various control instructions within the Philco Operating System aid the programmer during the debugging of his program. DUMP and SNAP instructions provide two methods of obtaining information stored within various parts of memory via the High Speed printer. Three other instructions, OCT, CMD, (COMMAND), and ALPHA, permit replacement of a word in memory in octal, command, and alphanumeric representation, respectively.

The post-mortem dumping of the DUMP control instruction takes place in three stages. The first, or pre-dump stage, which takes place whenever a DUMP control instruction is encountered by 32KSYS, stores information concerning the dump request within the dump table in memory. The second, or intermediate stage, copies binary data from memory to TSYSDMP. The third, or conversion phase, edits the memory specified by the dump table in the specified format and transfers the edited information to TSYSOUT.

The latter two stages will take place only if a post-mortem dump is initiated. (Refer to 1ZERO, 1ERRDMP, and 1SUBERR.) The dumping process for the SNAP control instruction differs from that of the DUMP instruction only in that the data from the location(s) to be snapped is transferred to TSYSDMP during the running of the program. The unedited data on TSYSDMP is edited to TSYSOUT if control is transferred to 1ZERO, 1ERRDMP, 1SUBERR, 1ENDJOB, 1NXTCON, 1XCONER or Location 1 (operator action only), or during a CLOBDMP procedure, provided TWYSDMP has not been rewound.

## DUMP

### DUMP - Dumps Memory

#### FUNCTION

Specifies what information is to be edited for the High-Speed Printer in a post-mortem dump.

#### FORMAT

L	Location	Command	Address and Remarks
		DUMP	<u>format</u> , <u>addr1</u> , <u>addr2</u>

#### PARAMETERS

format - The format in which information within the specified memory locations is to be dumped.

A Alphanumeric Conversion  
C Command Conversion  
F Floating Point Conversion  
H Hexadecimal Conversion  
O Octal Conversion  
nS Fixed Point Conversion

n (OPTIONAL) - A decimal number, 0 through 47, which indicates the position of the binary point. If n is omitted, 0 will be assumed.

addr1 - The starting address of the area to be dumped.  
(Refer to SYSOPS 8.)

addr2 - The ending address of the area to be dumped.  
(Refer to SYSOPS 8.)

#### REMARKS

1. If an illegal parameter is used, if any parameter is omitted, or if addr2 is less than addr1, the DUMP card is ignored.
2. A post-mortem dump occurs whenever control is transferred to 1ERRDMP, 1SUBERR, or 1ZERO.
3. DUMP control instructions should normally appear prior to the instruction which initiates the running of a program. Information specified by the parameters is saved by 32KSYS until a new JOB instruction is encountered.
4. A maximum of ten DUMP instructions may be specified. Any DUMP instruction in excess of ten will be ignored.

**DUMP**

5. Semi-colons or commas may be used interchangeably as break characters in the DUMP card.
6. DUMP cards containing errors are ignored and an appropriate error message is written on TSYSOUT.

**EXAMPLE ONE**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	DUMP		O, 11000, 17000

Explanation: Execution of this instruction causes the dump routine to be preset to execute a dump from location 11000<sub>8</sub> through 17000<sub>8</sub> in octal format.

**EXAMPLE TWO**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	DUMP		A, NAME.FLAD+33, 1400P

Explanation: Execution of this instruction causes the dump routine to be preset to execute a dump, from location NAME.FLAD+33 through the location represented by the sum of 1400<sub>8</sub> and the origin of the last REL program loaded, in alphanumeric format. This instruction must immediately follow the loading of the REL program which contains NAME.FLAD. NAME.FLAD is defined by the REL loader in the NAME/SYMBOL list if a TAC SYMBOUT instruction at compile time, or a 32KSYS SYMDEF instruction at run time, permits recognition of the symbol.

## SNAP

## SNAP - Snaps Memory

**FUNCTION**      Specifies what information is to be edited for the High-Speed Printer when designated conditions occur during the running of a program.

FORMAT	L	Location	Command	Address and Remarks
	<u>p</u>	<u>loc</u>	SNAP	<u>format;addr1;addr2;</u> <u>addr3;cond</u>

**PARAMETERS**      The parameters in the Address and Remarks field are separated by semi-colons.

p - L or R, which indicates whether the left or right instruction of a particular location will initiate the snap routine. The snap occurs prior to the execution of the instruction. If p is omitted L will be assumed.

loc - The location containing the instruction that will initiate the snap routine.

format - The format in which information within the specified memory locations is to be dumped.

A Alphanumeric Conversion  
C Command Conversion  
F Floating-point Conversion  
H Hexadecimal Conversion  
O Octal Conversion  
nS Fixed-point Conversion

n (OPTIONAL) - A decimal number, 0 through 47, which indicates the position of the binary point. If n is omitted, 0 will be assumed.

\*\*\*\*\*

addr1 - the location of the first word to be dumped.

addr2 - the location of the last word to be dumped.

addr3 (OPTIONAL) - The address of the last word in a six-word dump table. If no address is specified, the snap table will be stored in memory immediately preceding the last snap table. If no address is specified for any snap table, the first table is stored either at the end of memory (RPL) or (ABS), or immediately preceding the last program loaded (REL). It is the programmer's responsibility to select the appropriate locations and provide sufficient space for all his snapshot tables.

cond (OPTIONAL) - The condition which must be satisfied before a SNAP can be executed. If omitted, a snap dump will take place each time the instruction is to be executed.

There are two kinds of conditions:

1. The M/N Condition

A snapshot dump may be executed the Mth through the Nth time an instruction is executed where M and N are decimal numbers.

2. Comparison conditions

The contents of a memory location or a register are compared in an alphanumeric sense only, to the contents of a memory location, a register, or octal value.

The following comparisons can be made (a and b represent the contents of a memory location, a register or a value):

a = b	a equals b
a < b	a is less than b
a > b	a is greater than b
a ≠ b	a is not equal to b

The following rules govern the compared locations:

- a. Addresses of memory locations must be parenthesized.

Example: (125)<5

A snapshot dump will be given at the specified location when the contents of location 1258 are less than 5. (The value 5 is right justified with leading zeroes.)

- b. The notations A, Q, and D refer to the contents of the A Register, the Q Register and the D Register. The notation "n" (n=0, 1, 2--7) refers to the contents of index register n, right justified with leading zeros. When referring to the contents of a register, the parentheses must be omitted.

**SNAP****Examples**

$, 2 > 1$

A snapshot dump will be given at the specified location when the contents of index register 2 are greater than 1.

$A \# (15000)$

A snapshot dump will be given at the specified location when the contents of the A register are unequal to the contents of 150008.

- c. An address can be index modified.

**Example**

$(157, 4) > 0$

A snapshot dump will be given at the specified memory location when the contents of 157<sub>8</sub> modified by the contents of index register 4 are negative.

**\* NOTE****REMARKS**

1. The program to be run must be loaded into memory prior to the appearance of the SNAP control instruction(s). (SNAP calls may be compiled with a program through the use of the SNAPGEN generator.)
2. If p/loc is the same on two or more SNAP control instructions, the snap routines for those instructions are performed in the reverse order from which they were submitted. SNAP instructions which specify different addresses need not be sequenced by address.
3. When control is returned from the SNAP routine to the object program, all registers except JA are restored.
4. There is no limit to the number of SNAP instructions which can be given with each job, except that sufficient memory space should be available for snap tables (six locations per call).
5. The instruction at p/loc may not be a TIJ, TJM, TIO, SKC, or RPT. In addition, this instruction may not be under the influence of an RPT, TIO, or SKC instruction, nor may it be program-modified (e.g., by EIS).

or TJM). If one of these conditions is in effect (1) the return from the snapshot call may be to the wrong location in the object program (2) a TJM may give unexpected results, or (3) control may be transferred to location Zero.

#### 6. Illegal SNAP cards cause an exit to 1XCONER.

#### EXAMPLE ONE

This example illustrates the use of a conditional snapshot dump:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		JOB	XYZ
		RPL	4, ALPHA
L	15260	SNAP	0; 0, 3; 177, 3;; A#0
		JMP	*

Explanation: Execution of these instructions causes program ALPHA to be loaded and run. An octal dump of locations 0, 3 through 177, 3 will be performed before the execution of the instruction at 15260, if the contents of the A Register are non-zero. The SNAP table is situated in the last six memory locations at the end of memory.

#### EXAMPLE TWO

This example illustrates the use of an unconditional snapshot dump:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	14430	SNAP	F;14P;30P

Explanation: Execution of this instruction will cause an unconditional floating-point dump of the location represented by the sum of 14g and the origin of the last REL program loaded through the location represented by the sum of 30g and the origin of the last REL program loaded to be performed prior to executing the instruction at 14430L (L is assumed).

**OCT****OCT - Replaces Word(s) in Octal Format****FUNCTION**

Replaces the contents of one or more consecutive memory locations with words in octal format.

**FORMAT**

L	Location	Command	Address and Remarks
	<u>loc</u>	OCT	<u>word<sub>1</sub></u> ,..... <u>word<sub>n</sub></u>

**PARAMETERS**

loc - The location of the word to be changed.

word<sub>1</sub> - 16 or fewer octal digits which indicate the bit configuration to be inserted into the first location to be changed. If fewer than 16 characters are specified, the remainder of the word is right filled with zeros.

word<sub>n</sub> (OPTIONAL) - One or more additional parameters in the same format as word<sub>1</sub>, to be inserted into subsequent consecutive locations.

**REMARKS**

If corrections are to be made to relocatable programs, the following information should be noted:

1. The NAME/SYMBOL list may be stored in the TEMPORARY/ASTOR area of a program and if necessary at the area in memory starting at 10000g. However, the symbol list is cleared prior to executing the program.
2. Therefore the programmer should be especially careful when correcting the TEMPORARY/ASTOR area or the area starting at 10000g so as not to affect the symbol list and/or its connections. If the symbol list and/or its connections are affected, 32KSYS will err when clearing the symbol list prior to executing the program.
3. See explanation of PORIG if LOADSETS. AVAILMEM or any other modification of program origin has occurred prior to execution of CMD, ALPHA, or OCT corrections.

**EXAMPLE**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	100C	OCT	111111111111, 222222, 333
	200P	OCT	4444444444444444, 55

**Explanation:** Execution of these instructions will replace the contents of the memory locations listed below with the following information (expressed in Octal format).

<u>Location</u>	<u>New Contents</u>
Common + 100 <sub>8</sub>	12/001
Common + 101 <sub>8</sub>	6/010
Common + 102 <sub>8</sub>	3/011
Program origin + 200 <sub>8</sub>	16/100
Program origin + 201 <sub>8</sub>	2/101

**CMD**

**CMD (or COMMAND) - Replaces Program Instruction(s)  
in Command Format**

**FUNCTION**

Replaces the contents of one or more consecutive half-words in memory with instructions in command format.

**FORMAT**

L	Location	Command	Address and Remarks
<u>p</u>	<u>loc</u>	CMD	<u>instr1, addr1 ; .. instrn</u> <u>addrn</u>

**PARAMETERS**

p - Position of the first half-word to be replaced.

L (or blank)

Specifies the left instruction

R

Specifies the right instruction

loc - The location of the first word to be replaced.

instr1 addr1 - The new instruction and its address, separated by a comma, where:

instr1 is a TAC mnemonic, and

addr1 is an address. If indexed, the form must be:

val, i where val is an octal number from 0-7777  
and i is a number from 0-7.

instrn addrn (OPTIONAL) - One or more additional parameters in the same format as that for instr1, addr1. Parameters within the CMD control line must be separated by semi-colons.

**ACTION**

One or more consecutive half-words of memory, starting at the p half of loc, are replaced by one or more half-words designated by instr1, addr1;..... instrn, addrn.

**REMARKS**

1. CMD and COMMAND may be used interchangeably as the call for this control instruction.
2. If the address portion of a parameter is omitted, the address field of the half-word to be replaced will be filled with zeros.

3. If the address portion of a parameter is index register modified, the entire parameter must contain two commas, one between the instruction portion and val and one between val and i.
4. Any errors detected will cause control to be transferred to the system via 1XCONER.
5. Refer to OCT, REMARKS, concerning relocatable corrections.

## EXAMPLE ONE

This example illustrates the use of an absolute address correction.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
R	12120	CMD	TDXL, 0, 5;CM, 17677

Explanation: Execution of this instruction causes two half-words contained in memory locations 12120R, and 12121L to be replaced by the binary representation for TDXL 0, 5 CM 17677.

## CMD

### EXAMPLE TWO

This example illustrates the use of a relative address correction:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		REL	4, CORRECT \$
R	210P	CMD	JNOL, 14P; TMQ, 30C

Explanation: Execution of the first control instruction above causes the program CORRECT to be loaded without subroutines. If SUBS were specified, the origin used for the corrections would be that of the last subroutine loaded rather than that of the program CORRECT (refer to the Special Function PORIG). Execution of the second control instruction causes the two command corrections to be made, starting at the right half of the word at 2108 relative to the program origin. The first command operand address is relative to the program origin; the second to COMMON.

The results of the CMD instructions (expressed in TAC format) will be:

<u>Location</u>	<u>New Contents</u>
Program Origin + 2108 (right half)	JNOL, Program Origin + 148
Program Origin + 2118	TMQ, CSA of COMMON + 308

**ALPHA****ALPHA - Replaces Word(s) in Alphanumeric Format****FUNCTION**

Replaces the contents of one or more consecutive memory locations with words in alphanumeric format.

**FORMAT**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
<u>n</u>	<u>loc</u>	ALPHA	<u>replacement</u>

**PARAMETERS**

n (OPTIONAL) - The number (1-7) of words to be replaced. If omitted, 1 is assumed.

replacement - Sets of eight alphanumeric characters. Spaces are significant. All Philco characters are permitted.

**ACTION**

n - Consecutive words starting at loc are replaced by replacement.

**REMARKS**

Refer to OCT, REMARKS, concerning relocatable corrections.

Alpha replacements must start at the beginning of the Address and Remarks field.

**EXAMPLE**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
3	205P	ALPHA	ONEΔANDΔTWOΔARE
2	11200	ALPHA	THREE,ΔΔ \$ ABSOLUTEEXAMPLEΔ\$

Explanation: Execution of these instructions causes the contents of the memory locations listed below to be replaced with the following information (expressed in TAC format):

<u>Location</u>	<u>New Contents</u>
Program Origin + 2058	W/ONEΔANDΔ \$
Program Origin + 2068	W/TWOΔAREΔ\$
Program Origin + 2078	W/THREE,ΔΔ \$
112008	W/ABSOLUTE\$
112018	W/EXAMPLEΔ\$

## **SECTION XII**

### **SPECIAL FUNCTIONS**

This category of 32KSYS control instructions includes those which perform miscellaneous computer operations. Among these operations are filling memory, relaying messages to the operator via the Console Typewriter, and halting the computer.

## **CLEAR**

### **CLEAR - Clears Memory**

**FUNCTION**      Clears all memory from location  $10000_8$  to the end of memory to fixed point zeros (48 zero bits per word).

**FORMAT**

L	Location	Command	Address and Remarks
		CLEAR	

**ACTION**      All memory locations between  $10000_8$  and the effective end of memory inclusive are cleared to fixed point zeros.

## CLOCK

### CLOCK - Types Accounting Clock Date and Time on Console Typewriter

#### FUNCTION

Causes the date and time from the Accounting Clock to be typed on the Console Typewriter.

#### FORMAT

L	Location	Command	Address and Remarks
		CLOCK	

#### ACTION

The date and time as indicated by the Accounting Clock are typed on the Console Typewriter (refer to Normal Type-Out , below) in the form  $\Delta MM-DD \Delta HH-MM-T \Delta$  and stored in the Address and Remarks field of the CLOCK control card before the card is written on TSYSOUT.

#### REMARKS

1. If the CLOCK instruction is given and a computer has no Accounting Clock, the instruction will be ignored.
2. If the Accounting Clock is unavailable when a CLOCK command is given, the Error Type-Out, below, will occur and the next control instruction will be executed.

#### CONSOLE TYPEWRITER NORMAL TYPE-OUT

##### Type Out

##### Explanation

##### date, time

The CLOCK instruction was encountered and the date and time are typed via the Account Clock in the form:  $\Delta MM-DD \Delta HH-MM-T$  where MM-DD is the month and day, and HH-MM. T is the hour (per 24-hour day), the minute, and tenth of a minute.

#### CONSOLE TYPEWRITER ERROR TYPE-OUT

##### CLOCK FAILURE

The Accounting Clock was interrogated but was unavailable.

## COMMON

### COMMON - Sets Common Storage Area

#### FUNCTION

Sets the length and core starting address of the common storage area to be used by relocatable binary programs.

#### FORMAT

L	Location	Command	Address and Remarks
		COMMON	<u>length</u> , <u>m</u>

#### PARAMETERS

length - The number (decimal) of locations to be set aside for Common storage.

m - The starting address of the Common storage area. If a value less than  $10000_8$  is specified, exit will be made to 1XCONER.

#### ACTION

m and length are stored for use by the loader.

#### EXAMPLE

Assume that a relocatable binary program is to be loaded and run.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		COMMON	596, 12000

Explanation: Execution of this instruction causes the COMMON storage area to be preset to  $596_{10}$  locations, beginning at location  $12000_8$ .

## **ENDMEM**

### **ENDMEM - Sets the Effective End of Memory**

#### **FUNCTION**

Sets the effective end of memory for System functions.

#### **FORMAT**

L	Location	Command	Address and Remarks
		ENDMEM	<u>m</u>

#### **PARAMETER**

m - The new effective end of memory.

#### **ACTION**

The effective end of memory is set to m.

#### **REMARKS**

The effective end of memory is reset to 77777<sub>8</sub> when a JOB control instruction is executed.

## FILL

### FILL - Fills Memory

#### FUNCTION

Fills memory locations with the indicated filler.

#### FORMAT

L	Location	Command	Address and Remarks
		FILL	<u>addr</u> <sub>1</sub> , <u>addr</u> <sub>2</sub> , <u>filler</u>

#### PARAMETERS

addr<sub>1</sub> (OPTIONAL) - The starting address of the area in memory to be filled. If omitted, addr<sub>1</sub> is automatically set to 10000<sub>8</sub>.

addr<sub>2</sub> (OPTIONAL) - The ending address of the area in memory to be filled. If omitted addr<sub>2</sub> will automatically be set to the effective end of memory.

filler (OPTIONAL) - 0 to 16 octal characters.

1. If no characters, memory specified will be filled with zeros.
2. If two characters, memory specified will be filled with a word containing the 2 characters duplicated 8 times.
3. If more than two characters, memory specified will be filled with the characters converted and right justified.

#### REMARKS

If all parameters are omitted, memory from location 10000<sub>8</sub> to the effective end of memory will be filled with zeros.

#### ACTION

All memory locations between addr<sub>1</sub> and addr<sub>2</sub> inclusive are filled with filler.

**HLT - Halts Computer Operation****FUNCTION**

Relays messages to the operator via the Console Typewriter and stops the computer to permit a manual operation.

**FORMAT**

L	Location	Command	Address & Remarks
		HLT	<u>message</u>

**PARAMETER**

message - Fifty-six or fewer alphanumeric characters (including spaces) written in the Address and Remarks field, that compose a message to the operator.

**ACTION**

message is typed on the Console Typewriter. Upon completion of the type-out, the computer halts, enabling the operator to perform the manual action specified by message. Pressing advance causes 32KSYS to read and execute the next control instruction.

**REMARKS**

When the computer stops, all ones are displayed in the A, Q, and D Registers, indicating to the operator that this is a system halt.

## IBIT

### TRIT - Changes Indicator Bits

#### FUNCTION

Changes the bit configuration in the indicator word (1IBIT), which serves as a pseudo Toggle.

#### FORMAT

L	Location	Command	Address and Remarks
		IBIT	<u>n</u> , <u>n</u> , --- <u>n</u>

#### PARAMETER

n - One or more decimal numbers, 0 through 47 which designate the specific bit(s) to be changed (i.e., to change a zero to a one, or a one to a zero).

#### ACTION

- Each number placed in the Address and Remarks field as a parameter reverses its particular bit in 1IBIT.
- 1IBIT is cleared when a JOB instruction is executed.
- Bits in 1IBIT may be sensed by the ALTAC IF SENSE BIT statement.

**JMP - Transfers Control****FUNCTION**

Transfers control to a specified address.

L	Location	Command	Address and Remarks
		<u>jmp</u>	<u>addr</u>

**PARAMETERS**

jmp - A JMP command which indicates the portion of the word to which control is to be transferred.  
One of the following must be selected:

JMP Indicates the left half of the word

JMPL Indicates the left half of the word

JMPR Indicates the right half of the word

addr - The address to which control is to be transferred.

If an asterisk (\*) is specified, the jump address will be the starting address of the last program loaded.

**REMARKS**

JMP, JMPL and JMPR will have the same action if an asterisk is used as the parameter.

NOPRINT

**NOPRINT** - Inhibits Printing of Control Instructions

**FUNCTION** Sets 32KSYS to inhibit the printing of executed control instructions.

FORMAT	L	Location	Command	Address and Remarks
			NOPRINT	

## PARAMETERS

**ACTION** 32KSYS is set to inhibit the printing of executed control instructions until a PRINT or a JOB instruction is encountered.

## **PRINT**

### **PRINT - Permits Printing of Control Instructions**

#### **FUNCTION**

Sets 32KSYS to print executed control instructions.

#### **FORMAT**

L	Location	Command	Address and Remarks
		PRINT	

#### **PARAMETER**

None

#### **ACTION**

32KSYS is set to print executed control instructions until a NOPRINT instruction is encountered.

## PORIG

**PORIG - Sets the P-Relative Address in the COMMAND (CMD), OCT, and ALPHA Correction routines**

### FUNCTION

Sets the P-relative address in the COMMAND (CMD), OCT, and ALPHA corrective routines to the absolute value of the addr or to the origin of the id.

### FORMAT

L	Location	Command	Address and Remarks
		PORIG	<u>addr</u> or <u>id</u>

### PARAMETER

addr - The address to be used for P-relative corrections.

id - The identification of a program whose origin address will be used for P-relative corrections.

### ACTION

The address represented by the parameter in the command is placed in the P-relative address used by the COMMAND (CMD), OCT, and ALPHA correction routines.

## PROGTAPE - Assigns a Program Tape

## FUNCTION

Assigns a designated tape as the user's program tape.

## FORMAT

L	Location	Command	Address and Remarks
		PROGTAPE	<u>t</u>

## PARAMETER

t - The symbolic tape onto which subsequent compiled programs will be written.

## ACTION

All subsequent programs compiled within the current job will then be written on t following compilation.

## REMARKS

1. Any tape designated as the program tape must contain a block of Z's, which indicates the location for the storage of the next program.
2. A PROGTAPE assignment is valid until another PROGTAPE or JOB control card.
3. t may not be a system tape.
4. Writes block of Z's when done.

## **REM**

### **REM - Relays Remarks to an Operator**

#### **FUNCTION**

Relays messages to the operator via the Console Typewriter without stopping computer action.

#### **FORMAT**

L	Location	Command	Address and Remarks
		REM	<u>message</u>

#### **PARAMETER**

message - Fifty-six or fewer alphanumeric characters (including spaces) written in the Address and Remarks field, that compose a message to the operator. The characters should be acceptable to the Console Typewriter. (Refer to 1TYPOUT, SYS Entries).

#### **ACTION**

message is typed on the Console Typewriter without stopping computer action.

**SYSOPS - Compiler Option Word****FUNCTION**

Changes bits in the compiler option word (1SYSOPS) to offer the programmer additional compiler options.

**FORMAT**

L	Location	Command	Address and Remarks
		SYSOPS	<u>n, n, n, ...n</u>

**PARAMETER**

n - One or more decimal numbers, 0 through 10, which designate the specified bit to be changed.

**ACTION**

When a SYSOPS control instruction is executed, the designated bits are reversed in 1SYSOPS for use by 32KSYS or compilers.

**REMARKS**

1. 1SYSOPS is cleared whenever a new JOB instruction is executed.
2. Exit will be made to 1XCONER if any number greater than 10 is specified in the Address and Remarks field or the SYSOPS control instruction.

The table on the following page describes the bit location within 1SYSOPS that may be changed at the programmer's discretion:

## SYSOPS

<u>Bit</u>	<u>Explanation</u>
0	If set, Philco 212 mnemonic commands will be compiled into their proper bit configuration, otherwise such commands will be considered command faults.
1	If set, POSSIBLE F-BIT ERROR will appear on the code edit whenever a possible F-Bit error is detected during a TAC compilation.
2	If set, TAC will not store the intermediate code in the higher portion of memory. Instead, the code will be written onto TSYSCR3, and the memory space released will be used for the symbol table.
3	If set, the compiler assumes that its input will be in binary-image mode on a tape other than TSYSIN.
4	If set, the compiler assumes that its input will be in hollerith-image mode on a tape other than TSYSIN.
5	If set, the COBOL compiler interprets columns 1-72 only and also considers the 8-4 punch as the quote character rather than the semi-colon.
6	If set, the COBOL compiler operates in "test mode" to aid in compiler debugging. Intermediate compiler files are edited and added to the normal compiler printer output.
7	If set, the internal loader (LINTLD) will search the tape with the symbolic tape name XSYSTEM. (This bit is tested only when the internal loader is instructed to by the object program).
8	If set, the <u>addr<sub>1</sub></u> and/or <u>addr<sub>2</sub></u> parameters of DUMP control instructions less than 10000 <sub>8</sub> will be valid, and will permit the programmer to obtain dumps of the memory area reserved for 32KSYS.
9	If set, the FORTRAN IV compiler operates in "test mode" to aid in compiler debugging; i.e., IBITS 0-18 will be interrogated by the FORTRAN IV compiler.
10	If set, all 32KSYS preset symbols are inhibited, enabling 8KSYS programs not requiring 8K tape libraries at compile time to be compiled with 32KSYS TAC.

**SYMDEF - Defines a Symbolic Address****FUNCTION**

Adds the defined symbolic address to the SYMBOUT list, and defines REFOUT's having symbolic addresses identical to it.

**FORMAT**

L	Location	Command	Address and Remarks
		SYMDEF	<u>symbol</u> <sub>1</sub> , <u>m</u> <sub>1</sub> ; ..... <u>symbol</u> <sub>n</sub> , <u>m</u> <sub>n</sub>

**PARAMETERS**

symbol - The symbol (NAME.FLAD) being defined.

m - The address which defines symbol.

**ACTION**

An attempt is made to find symbol in the REFOUT list. If found, m is used to satisfy loading conditions. Whether found or not, symbol and m are then added to the SYMBOUT list for subsequent reference.

**REMARKS**

If symbol is greater than 16 characters, or NAME.FLAD is not separated by a period, or m is not an octal address, or each symbol is not defined by an address parameter, exit is made to 1XCONER.

Each symbol must be separated from m with a comma (,) and each symbol, m must be separated from the following symbol, m with a semi-colon (;).

## **SECTION XIII**

### **32KSYS ENTRIES**

32KSYS Entries permit the programmer to make use of specialized routines used by 32KSYS. The entries are locations within 32KSYS which contain a transfer of control to a specific routine. Some routines, at the conclusion of their operation, return control to the calling program, while others transfer control to another 32KSYS entry.

The programmer may use these routines by writing a transfer of control to the desired 32KSYS entry. This transfer must be made to a symbolic address as designated herein. This symbolic address will be defined automatically by TAC when the program is compiled.

## 1ADCONI

**IADCONI - Converts the First Parameter of a Control Instruction to an Octal Address**

FUNCTION	Converts the first parameter in the Address and Remarks field of a control instruction to an octal address.
SYMBOLIC ADDRESS	SYS.1ADCONI
PREDOMINANT USER	32KSYS and/or the programmer
PRESET REQUIREMENTS	The parameter must be an asterisk (*) [ 1NTRYJA will be substituted] or an address.
ACTION	<ul style="list-style-type: none"><li>• The first parameter in the Address and Remarks field of a control instruction is converted and stored in the A Register. Spaces are suppressed.</li><li>• Termination of the first parameter is determined by the appearance of a comma, dollar sign, slash, semi-colon, "equal" sign, "greater than" sign, right or left parentheses or number sign. The termination character is stored in 1BRKSTR right justified with leading zeros.</li><li>• If an error is detected, exit is made to 1XCONER,</li></ul>
EXIT	Control is returned to the next sequential instruction in the calling program.

## **1ADCONC**

**1ADCONC - Converts the Next Parameter of a Control Instruction to an Octal Address**

<b>FUNCTION</b>	Converts the next parameter in the Address and Remarks field of a control instruction to an octal address.
<b>SYMBOLIC ADDRESS</b>	SYS.1ADCONC
<b>PREDOMINANT USER</b>	32KSYS and/or the programmer
<b>PRESET REQUIREMENTS</b>	SAME AS 1ADCONI (refer to 1ADCONI)
<b>ACTION</b>	SAME AS 1ADCONI (refer to 1ADCONI)
<b>EXIT</b>	Control is returned to the next sequential instruction in the calling program.

## **1CLOCK**

### **1CLOCK - Obtains and Edits the Clock Word**

<b>FUNCTION</b>	Obtains and edits the clock word.
<b>SYMBOLIC ADDRESS</b>	SYS.1CLOCK
<b>PREDOMINANT USER</b>	32KSYS and/or the programmer
<b>PRESET REQUIREMENTS</b>	None
<b>ACTION</b>	The clock word is edited in the following form:  A Register = ΔΔMM-DDΔ Q Register = HH-MM.TΔ  where    MM = Month DD = Day HH = Hour MM = Minute T   = Tenth of minute
<b>EXIT</b>	Control is transferred to the next sequential instruction in the calling program.

## 1ENCODE

### 1ENCODE - Image to Code Conversion

FUNCTION	Converts an Image Mode card of $2n$ consecutive core memory locations into the corresponding Code Mode card of $n$ consecutive core memory locations.
SYMBOLIC ADDRESS	SYS.1ENCODE
PREDOMINANT USER	32KSYS and/or the programmer
PRESET REQUIREMENTS	The A-register should contain a word with bits 0 = 0 1-15 = IMAGE 24 = 0 25-39 = CODE 40 = 0 41-47 = n (if bits 41-47 are zero, n is set equal to 1).
	IMAGE is defined as the address of the first Image Mode word and CODE is defined as the address of the first Code Mode word.
ACTION	The Image Mode card is converted into the corresponding Code Mode card with each illegal Image Mode character represented in Code Mode by a question mark. The Code Mode characters are defined in Philco 2000 Code Combination Card, TF-17.
EXIT	Control is returned to the next sequential instruction in the calling program.
REMARKS	If the Image and Code areas overlap, then CODE (the address of the first code) must not fall between IMAGE + 2 through IMAGE + $2n$ - 1 inclusive.

## 1ENDJOB

### 1ENDJOB - End of Job Routine

FUNCTION	Terminates a job.
SYMBOLIC ADDRESS	SYS.1ENDJOB
PREDOMINANT USER	32KSYS and/or the programmer
ACTION	<ul style="list-style-type: none"><li>• Snaps are edited if any were written.</li><li>• A search for the next JOB control instruction is initiated.</li></ul>
	When a JOB control instruction is found: <ol style="list-style-type: none"><li>1. All tapes except those assigned to system PUN's and those held for the next job are rewound with lockout</li><li>2. All symbolic tape names except system symbolic tape names and those held for the next job are released</li><li>3. An end of job message is written on TSYSOUT</li><li>4. TSYSDMP is rewound</li><li>5. The JOB control instruction is executed.</li></ol>

## 1ERRDMP - Error Dump Routine

FUNCTION	Initiates 32KSYS post-mortem dumping.
SYMBOLIC ADDRESS	SYS.1ERRDMP
PREDOMINANT USER	The programmer
ACTION	<p>Whenever control is transferred to the 32KSYS post-mortem dump routine, either via 1ERRDMP, 1ZERO or 1SUBERR, the following action takes place:</p> <ul style="list-style-type: none"> <li>• The contents of all addressable registers are saved.</li> <li>• A type-out occurs on the Console Typewriter, signifying that a post-mortem dump is to be performed. A zero, 2, or 3 within the type-out indicates the memory location to which control was transferred.</li> <li>• If dumps are specified, snaps were written, or an error message is indicated, the dump edit routine is called.</li> <li>• TSYSDMP is used as an intermediate tape by the dump edit routine.</li> <li>• A search for the next JOB control instruction is initiated.</li> </ul> <p>When a JOB control instruction is found:</p> <ol style="list-style-type: none"> <li>1. All tapes except those assigned to system PUN's and those held for the next job are rewound with lockout</li> <li>2. All symbolic tape names except system symbolic tape names and those held for the next job are released</li> <li>3. An end of job message is written on TSYOUT</li> <li>4. TSYSDMP is rewound</li> <li>5. The JOB control instruction is executed.</li> </ol>

## **1ERRDMP**

<u>Console</u>	<u>Type-Out</u>	<u>Explanation</u>
TYPEWRITER		
ERROR		
TYPE-OUT	ERRDMP2	A programmed transfer of control to 1ERRDMP

**1GETBL - Gets Blanks During Scanning**

<b>FUNCTION</b>	Presets the 1SCANON subroutine to recognize (not suppress) spaces in the parameters contained in the Address and Remarks field of a control instruction.
<b>SYMBOLIC ADDRESS</b>	SYS.1GETBL
<b>PREDOMINANT USER</b>	32KSYS and/or the programmer
<b>PRESET REQUIREMENTS</b>	None
<b>ACTION</b>	1SCANON is set to recognize spaces.
<b>EXIT</b>	Control is returned to the next sequential instruction in the calling program.
<b>REMARKS</b>	1GETBL has no effect on the 1SCAN subroutine because 1SCAN automatically resets itself to suppress spaces each time it is executed.
<b>EXAMPLE</b>	Refer to 1SCANON for examples using the five scan subroutines.

## IIGBL

### IIGBL - Ignore Blanks During Scanning

FUNCTION	Presets the ISCANON subroutine to suppress spaces in the parameters contained in the Address and Remarks field of a control instruction.
SYMBOLIC ADDRESS	SYS.IIGBL
PREDOMINANT USER	32KSYS and/or the programmer
PRESET REQUIREMENTS	None
ACTION	ISCANON is set to suppress spaces.
EXIT	Control is returned to the next sequential instruction in the calling program.
REMARKS	IIGBL has no effect on the ISCANBL subroutine because ISCANBL automatically resets itself to recognize spaces each time it is executed.
EXAMPLE	Refer to ISCANON for examples using the five scan subroutines.

## 1INTCMD - Executes a 32KSYS Control Instruction

FUNCTION	Executes a 32KSYS control instruction.
SYMBOLIC ADDRESS	SYS.1INTCMD
PREDOMINANT USER	The programmer
PRESET REQUIREMENTS	Index register 7 contains the address of the first word to be loaded into 1CONLIN+1.
ACTION	<ul style="list-style-type: none"> <li>• 1CONLIN+1 to 1CONLIN+9 is loaded.</li> <li>• The 32KSYS control instruction in 1CONLIN +1 to 1CONLIN +9 is executed.</li> <li>• If an error occurs while executing the 32KSYS control instruction, exit is made to 1XCONER.</li> <li>• Index register 7 contains, on exit, the address of the last word transferred+1.</li> </ul>
REMARKS	The format of the control instruction may be in Console Typewriter format (See Operating Procedures, Special Operator Actions) or standard 32KSYS card format.
EXIT	Control is returned to the next sequential instruction in the calling program.

## 1INTCON

1INTCON - Loads 1CONLIN + 1 to 1CONLIN + 9

FUNCTION	Loads 1CONLIN+1 to 1CONLIN+9 with specified memory.
SYMBOLIC ADDRESS	SYS. 1INTCON
PREDOMINANT USER	The programmer
PRESET REQUIREMENTS	Index register 7 contains the address of the first word to be loaded into 1CONLIN+1.
ACTION	<ul style="list-style-type: none"><li>• 1CONLIN+1 to 1CONLIN+9 is loaded from internal memory.</li><li>• Index register 7 contains on exit the address of the last word transferred+1.</li></ul>
EXIT	Control is returned to the next sequential instruction in the calling program.
REMARKS	See also 1EXECUTI.

**1INTLD - Internal Loader**

<b>FUNCTION</b>	Loads a program
<b>SYMBOLIC ADDRESS</b>	SYS.1INTLD
<b>PREDOMINANT USER</b>	32KSYS and/or the programmer
<b>PRESET REQUIREMENTS</b>	The A and Q Registers contain:  1. The identification <u>id</u> of the program to be loaded in the <u>A Register</u> . The <u>id</u> must be eight characters or fewer, left-justified and space-filled to the right if necessary.  2. The following specified bit designations are to be placed in the <u>Q Register</u> :
Bits 18-23	The program unit number (PUN) on the tape on which the program is stored.
Bit 43	If set to one, bit 7 of 1SYSOPS will be checked for loading from an alternate system tape.
Bit 44	If set to one, control is to be transferred to the starting address of the program after it is loaded.
Bit 45	If set to one, the direction of the search will be backward.
Bit 46	If set to one, an ABS program is to be loaded.
Bit 47	If set to one, an RPL program is to be loaded.

**CAUTION:** If Bits 46 and 47 are zero, an exit will be made to IXCONER. If more than one of these bits is given, bit 47 will be assumed to be the correct bit.

## 1INTLD

### ACTION

- If Bits 43 and 45 are zero, the tape specified by the Q Register will be searched forward for the program specified by id. If the program is not found before encountering a block of Z's the tape will be searched backward. If the program still cannot be found before encountering "begin tape", exit will be made to 1XCONER.
- If bit 45 is one, the tape specified will be searched backward only.
- If bit 43 is one and bit 7 of 1SYSOPS is one, a tape with the symbolic name XSYSTEM, rather than the PUN in the Q Register, will be searched.
- Control will be transferred to the loaded program if bit 44 of the Q Register is set to one, otherwise control is returned to the next sequential instruction in the calling program.

### REMARKS

1. Internal loader calls may be inserted in a program during compilation by use of the internal load generator, LOADGEN (Library routines).

### EXAMPLE

The following example illustrates the calling of a program into memory by another program during the latter's execution.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
----------	-----------------	----------------	----------------------------

TMA	W/SEG1	\$
TMQ	N/6 T23; 1/1	T47\$
JMP	SYS.1INTLD	

Explanation: Execution of these instructions causes the program SEG1 to be loaded into memory and control returned to the next sequential instruction in the calling program.

## **1MEMSIZ**

### **1MEMSIZ - Checks Memory Overflow**

FUNCTION	Tests the given address for memory overflow.
SYMBOLIC ADDRESS	SYS.1MEMSIZ
PREDOMINANT USER	32KSYS and/or the programmer
PRESET REQUIREMENTS	The A Register contains the address to be tested at T39.
ACTION	<ul style="list-style-type: none"><li>● The specified address is tested against the effective end of memory.</li><li>● If the address is legal, the Q Register is made zero.</li><li>● If the address is illegal, the Q Register is made all ones.</li><li>● The A Register contains the effective end of memory +1 at T39.</li></ul>
EXIT	Control is returned to the next sequential instruction in the object program.
REMARKS	The effective end of memory, reset by the JOB control instruction to $77777_8$ , can be changed by the ENDMEM control instruction.

## 1NXTCRD

## 1NXTCRD - Obtains the Next Card

FUNCTION	Obtains from TSYSIN the next card in Code Mode (converting from Image, if necessary) and stores it in the ten memory locations beginning at 1CONLIN.
SYMBOLIC ADDRESS	SYS.1NXTCRD
PREDOMINANT USER	32KSYS and/or the programmer
PRESET REQUIREMENTS	If Bit 43 of 1CONBIT is set to one and the mode of input is image, only the command field will be converted to Code Mode and transferred to 1CONLIN+2. Other data in 1CONLIN remain unchanged.
ACTION	<ul style="list-style-type: none"> <li>● The location of the first word of the next card, which was read into an input buffer area in memory as part of a block from the system input tape is stored in Index Register 1.</li> <li>● If input is in Code Mode, the ten words in the buffer area which compose the next card will be transferred directly to 1CONLIN.</li> <li>● If input is in Image Mode, it will automatically be converted to Code Mode during its transfer to 1CONLIN (subject to Bit 43 of 1CONBIT), and the location of the image card in the buffer area will be placed into Index Register 1. Illegal Image Mode characters appear as question marks (?) in Code Mode.</li> </ul>
EXIT	Control is returned to the next sequential instruction in the calling program.
REMARKS	If a JOB card is encountered, an exit will be made to 1XCONER.

**1NXTCON - Next Control Instruction Routine**

FUNCTION	Obtains and executes the next control instruction.
SYMBOLIC ADDRESS	SYS.1NXTCON
PREDOMINANT USER	32KSYS and/or the programmer
ACTION	<ul style="list-style-type: none"><li>• Snaps are edited, if any were written.</li><li>• The next control instruction is obtained from the ten memory locations starting at 1CONLIN via the 1NXTCRD routine.</li><li>• When the instruction is obtained, the command field is interpreted.</li><li>• The NAME/SYMBOL list is cleared if any had been established.</li></ul>
EXIT	If the control instruction is legal, it will be executed. Illegal instructions cause a transfer to 1XCONER.

**1PRTCON**

**1PRTCON - Writes 1CONLIN on TSYSOUT**

FUNCTION	Writes 1CONLIN to 1CONLIN+9 on TSYSOUT
SYMBOLIC ADDRESS	SYS.1PRTCON
PREDOMINANT USER	32KSYS and/or the programmer
ACTION	1CONLIN to 1CONLIN+9 is edited for printing (data select 0, line feed 1, and end of line character) and is written on TSYSOUT.
EXIT	Control is returned to the next sequential instruction in the calling program.

**1PRTMSG - Writes Message on TSYSOUT**

<b>FUNCTION</b>	Writes message on TSYSOUT
<b>SYMBOLIC ADDRESS</b>	SYS.1PRTMSG
<b>PREDOMINANT USER</b>	32KSYS and/or the programmer
<b>ACTION</b>	The fifteen words beginning at 1MSGCON are edited for printing (data select 0, line feed 1, and end of line character) and are written on TSYSOUT.
<b>EXIT</b>	Control is returned to the next sequential instruction in the calling program.

## **1PRTRUN**

### **1PRTRUN - Empties Buffer for TSYSOUT**

<b>FUNCTION</b>	<b>Empties buffer for TSYSOUT</b>
<b>SYMBOLIC ADDRESS</b>	<b>SYS.1PRTRUN</b>
<b>PREDOMINANT USER</b>	<b>32KSYS and/or the programmer</b>
<b>ACTION</b>	<b>The output buffer for TSYSOUT is emptied.</b>
<b>EXIT</b>	<b>Control is returned to the next sequential instruction in the calling program.</b>

**1PUNSYM - Finds Symbolic Tape Name**

FUNCTION	Finds a symbolic tape name, referencing a given program unit number (PUN).
SYMBOLIC ADDRESS	SYS.1PUNSYM
PREDOMINANT USER	32KSYS and/or the programmer.
PRESET REQUIREMENTS	The A Register contains a PUN at T23.
ACTION	If the PUN is illegal or undefined, exit is made to 1XCONER.  If the PUN is legal, the first symbol in 32KSYS's symbolic tape name list assigned the given PUN is left in the A Register left-justified with trailing blanks.
EXIT	Control is returned to the next sequential instruction in the calling program.

## **1SCAN**

### **1SCAN - Scans the First Parameter of a Control Instruction Suppressing Spaces**

<b>FUNCTION</b>	Initializes the 32KSYS scanning subroutine and stores the first parameter contained in the Address and Remarks field of a control instruction.
<b>SYMBOLIC ADDRESS</b>	SYS.1SCAN
<b>PREDOMINANT USER</b>	32KSYS and/or the programmer
<b>PRESET REQUIREMENTS</b>	None
<b>ACTION</b>	<ul style="list-style-type: none"><li>● The first parameter in the Address and Remarks field of a control instruction is scanned and stored in 1WRD1 and 1WRD2. Spaces are always suppressed.</li><li>● Termination of the first parameter is determined by the appearance of a standard 32KSYS break character (a comma, semi-colon, slash, period, or dollar sign).</li><li>● All characters of the first parameter (a maximum of 16) are stored upon exit from the subroutine in 1WRD1 and 1WRD2, acting as a double-length register. The characters are stored right-justified with leading zeros.</li><li>● The number of characters recognized by the 1SCANBL, 1SCAN, 1SCANON subroutines is stored in the left-address portion of 1CHARCT prior to the exit from the subroutine.</li><li>● Because the subroutine stores 16 or fewer characters, the results are not guaranteed if more than 16 are used.</li></ul>
<b>EXIT</b>	Upon completion of the subroutine, control is returned to the next sequential instruction in the calling program.

## **1SCAN**

### **REMARKS**

1. 1SCAN always suppresses spaces within the first parameter.
2. If additional parameters of the same control line are to be scanned, 1SCANON must be used.

### **EXAMPLE**

Refer to 1SCANON for examples using the five scan subroutines.

## 1SCANBL

**1SCANBL      Scans the First Parameter of a Control Instruction Recognizing Spaces**

FUNCTION	Initializes the 32KSYS scanning subroutine and stores the first parameter contained in the Address and Remarks field of a control instruction. Spaces are recognized (not suppressed).
SYMBOLIC ADDRESS	SYS.1SCANBL
PREDOMINANT USER	32KSYS and/or the programmer
ACTION	<ul style="list-style-type: none"><li>The <u>first</u> parameter in the Address and Remarks field of a control instruction is scanned and stored in 1WRD1 and 1WRD2. Spaces are relevant.</li><li>Termination of the first parameter is determined by the appearance of a standard 32KSYS break character (a comma, semi-colon, slash, period or dollar sign). This break character appears in the A Register and SYS.1BRKSTR at exit, right-justified with leading zeros.</li><li>All characters of the first parameter (a maximum of 16) are stored in 1WRD1 and 1WRD2, acting as a double-length register. The characters are stored right-justified with leading zeros.</li><li>The number of characters recognized by the 1SCANBL, 1SCAN, and 1SCANON subroutines is stored in the left address portion of 1CHARCT just prior to the exit from the subroutine.</li><li>Because the subroutine stores 16 or fewer characters, the results are not guaranteed if more than 16 are used.</li></ul>
EXIT	Upon completion of the subroutine, control is returned to the next sequential instruction in the calling program.
REMARKS	If additional parameters of the same control line are to be scanned, 1SCANON must be used.
EXAMPLE	Refer to 1SCANON for examples using the five scan subroutines.

## **1SCANON**

### **1SCANON - Scans Additional Parameters of a Control Instruction**

<b>FUNCTION</b>	Continues the scanning process begun by 1SCANBL or 1SCAN by storing the next parameter contained in the Address and Remarks field of a control instruction.												
<b>SYMBOLIC ADDRESS</b>	SYS.1SCANON												
<b>PREDOMINANT USER</b>	32KSYS and/or the programmer												
<b>ACTION</b>	The same action listed for 1SCAN and 1SCANBL, except that the parameter beginning at the first column following the last break character encountered is scanned and stored. The use of 1SCANBL or 1SCAN <u>must</u> precede the use of 1SCANON otherwise unpredictable results will occur.												
<b>REMARKS</b>	<ol style="list-style-type: none"><li>1. Additional 1SCANON calls must be given for subsequent parameters to be scanned.</li><li>2. 1SCANON is always initially set by 1SCANBL to recognize spaces or by 1SCAN to suppress spaces.</li><li>3. The automatic setting of 1SCANON to suppress or recognize spaces continues through each subsequent use of 1SCANON until changed by a 1GETBL or 1IGBL call.</li></ol>												
<b>EXAMPLE 1</b>	Assume that the Address and Remarks field of a control line stored in 1CONLIN contains the following characters:  Δ5, 0ΔBCΔ/ DΔE FΔΔΔΔΔG\$												
	<table><thead><tr><th><u>L</u></th><th><u>Location</u></th><th><u>Command</u></th><th><u>Address and Remarks</u></th></tr></thead><tbody><tr><td>.</td><td>.</td><td>.</td><td>.</td></tr><tr><td></td><td>JMP</td><td></td><td>SYS.1SCAN</td></tr></tbody></table>	<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>	.	.	.	.		JMP		SYS.1SCAN
<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>										
.	.	.	.										
	JMP		SYS.1SCAN										

**Explanation:** Execution of this instruction causes the first parameter (terminated by the comma) to be scanned and characters Δ5 within it to be placed in

## 1SCANON

1WRD1 as W/00000000  
1WRD2 as W/00000005

The break character (,) is placed in the A Register and  
1BRKSTR as

W/0000000,

The number of characters (1) is placed in 1CHARCT as

C/HLT, 1

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		JMP	SYS.1SCANON

Explanation: Execution of this instruction causes the second parameter (terminated by a slash) to be scanned and the characters within it (0ΔBCΔ) to be placed in

1WRD1 as W/00000000  
1WRD2 as W/000000BC

The break character (/) is placed in the A Register and  
1BRKSTR as

W/0000000/

The number of characters (3) is placed in 1CHARCT as

C/HLT, 3

Note that leading zeros must be determined by the character count.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		JMP	SYS.1GETBL
		JMP	SYS.1SCANON

Explanation: Execution of these instructions causes the third parameter (terminated by a dollar sign) to be

scanned and the characters within it (DΔEFΔΔΔΔG) to be placed in

1WRD1 as W/000000DΔ  
1WRD2 as W/EFΔΔΔΔG

The break character (\$) is placed in the A Register and 1BRKSTR as

W/0000000\$

The number of characters (10) is placed in 1CHARCT as

C/HLT, 10

#### EXAMPLE TWO

Assume that the Address and Remarks field of a control line stored in 1CONLIN contains the following characters:

Δ5, 0ΔBCΔ/ DΔEFΔΔΔΔΔG\$

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
.	.	.	.
	JMP	SYS.	1SCANBL

Explanation: Execution of this instruction causes the first parameter (terminated by a comma) to be scanned and characters, Δ5, within it to be placed in

1WRD1 as W/00000000  
1WRD2 as W/000000Δ5

The break character (,) is placed in the A Register and 1BRKSTR as

W/0000000,

The number of characters (2) is placed in 1CHARCT as

C/HLT, 2

## 1SCANON

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		JMP	SYS.1SCANON

Explanation: Execution of this instruction causes the second parameter (terminated by a slash) to be scanned and the characters within it (0ΔBCΔ) to be placed in

1WRD1 as W/00000000  
1WRD2 as W/0000ΔBCΔ

The break character (/) is placed in the A Register and 1BRKSTR as

W/0000000/

The number of characters (5) is placed in 1CHARCT as

C/HLT, 5

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		JMP	SYS.1IGBL
		JMP	SYS.1SCANON
		.	.
		.	.

Explanation: Execution of these instructions causes the third parameter (terminated by a dollar sign) to be scanned and the characters within it (DΔEFΔΔΔΔΔG) to be placed in

1WRD1 as W/00000000  
1WRD2 as W/0000DEFG

The break character (\$) is placed in the A Register and 1BRKSTR as

W/0000000\$

The number of characters (4) is placed in 1CHARCT as

C/HLT, 4

**1STRIPQ - Strips the Q Register of Spaces**

<b>FUNCTION</b>	Suppresses space characters in a word.
<b>SYMBOLIC ADDRESS</b>	SYS.1STRIPQ
<b>PREDOMINANT USER</b>	32KSYS and/or the programmer
<b>PRESET REQUIREMENTS</b>	The word to be stripped is in the Q Register.
<b>ACTION</b>	All spaces are deleted from the word in the Q Register.  The stripped word is then transferred to the A Register, right-justified with leading zeros.
<b>EXIT</b>	Control is returned to the next sequential instruction in the calling program.

**1SUBERR****1SUBERR - Subroutine Error Exit**

FUNCTION	Initiates 32KSYS post-mortem dumping.	
SYMBOLIC ADDRESS	SYS.1SUBERR	
PREDOMINANT USER	Philco 2000 Subroutines and/or the programmer.	
ACTION	The contents of the JA Register are saved. Control is transferred to the same 32KSYS post-mortem dump routine as that used by 1ERRDMP. Actions, except the type-out, listed for 1ERRDMP apply to 1SUBERR.	
REMARKS	If the programmer wishes, he may replace the automatic post-mortem dump entrance in 1SUBERR with an entrance to his own routine. 1SUBERR is reset to the standard 32KSYS function by the execution of a JOB control instruction.	
CONSOLE TYPEWRITER TYPE-OUT	<u>Type-Out</u>  ERRDMP3	<u>Explanation</u>  A programmed transfer of control to 1SUBERR.

**1TAPENO - Obtains a Program Unit Number (PUN)**

<b>FUNCTION</b>	Obtains a program unit number (PUN) for the given parameter.
<b>SYMBOLIC ADDRESS</b>	SYS.1TAPENO
<b>PREDOMINANT USER</b>	32KSYS and/or the programmer
<b>PRESET REQUIREMENTS</b>	The A Register contains the input parameter left justified with trailing blanks.  The parameter is either:
	<ol style="list-style-type: none"> <li>1. A symbolic tape name (a TAC symbol of 7 or fewer characters) or,</li> <li>2. A system PUN in Binary Coded Decimal.</li> </ol> <ul style="list-style-type: none"> <li>• The parameter is checked for legality.</li> <li>• If the parameter is illegal, exit will be made to 1XCONER.</li> <li>• If the parameter is a PUN, the A Register will contain the PUN (in binary) at T23 on exit.</li> <li>• If the parameter is a symbolic tape name, the symbolic tape name list will be searched. If the name is there, its associated PUN will be in the A Register at T23 on exit. If the name is not there, it is inserted in the list, a PUN is assigned, and that assigned PUN will be in the A Register at T23 on exit.</li> </ul>
<b>ACTION</b>	
<b>REMARKS</b>	At entrance, if the last character in the A Register is a slash (/), octal 61, instead of a blank, 32KSYS legalizes the assigned PUN for writing.
<b>EXIT</b>	Control is returned to the next sequential instruction in the calling program.

## 1TYPOUT

### 1TYPOUT - Console Typewriter Type-Out

FUNCTION	Types alphanumeric characters from memory on the Console Typewriter.
SYMBOLIC ADDRESS	SYS.1TYPOUT
PREDOMINANT USER	32KSYS and/or the programmer
PRESET REQUIREMENTS	The A Register should contain: <u>C/HLT, m;C/HLT, n</u> where <u>m</u> is the address of the first of <u>n</u> words to be typed on the Console Typewriter.
EXIT	Control is returned to the next sequential instruction in the calling program.
REMARKS	<ol style="list-style-type: none"><li>1. Trailing full words of spaces are not transmitted to the Console Typewriter.</li><li>2. The maximum number of characters per line of typeout is limited to the length of the Console Typewriter line (approximately 70 characters). Any number of lines may be typed using the carriage return character (octal 32) when appropriate.</li></ol>

## 1XCONER - Control Line Error Routine

FUNCTION	Causes an error message to be written on TSYSOUT and 32KSYS post-mortem dumping to be initiated.																								
SYMBOLIC ADDRESS	SYS.1XCONER																								
USER	<u>32KSYS only</u>																								
PRESET REQUIREMENTS	None																								
ACTION	Whenever control is transferred to 1XCONER, the following action takes place: <ul style="list-style-type: none"> <li>• An error indication is typed on the console type-writer (See ERROR MESSAGES).</li> <li>• The contents of the JA Register are stored in the left address portion of 1NTRYJA.</li> <li>• An error message is written on TSYSOUT. (See ERROR MESSAGES.)</li> </ul>																								
EXIT FROM ROUTINE	Following the type-out, control is transferred to the 32KSYS post-mortem dump routine (refer to 1ERRDMP).																								
REMARKS	Those 32KSYS error messages that are flagged by an asterisk in the following table inhibit the 32KSYS post-mortem dumping routine from producing dumps, even though requested by DUMP control instructions.																								
ERROR MESSAGES	<table border="0"> <thead> <tr> <th><u>Type-Out</u></th> <th><u>Printout</u></th> </tr> </thead> <tbody> <tr> <td>SYSE1</td> <td>PROGRAM NOT ON SPECIFIED TAPE</td></tr> <tr> <td>*SYSE2</td> <td>CHECKSUM ERROR</td></tr> <tr> <td>SYSE3</td> <td>PROGRAM ORIGIN IN LIST AREA</td></tr> <tr> <td>SYSE4</td> <td>PROGRAM ORIGIN IN COMMON AREA</td></tr> <tr> <td>SYSE5</td> <td>UNDEFINED REFOUT(S)</td></tr> <tr> <td>SYSE6</td> <td>ILLEGAL CARD</td></tr> <tr> <td>SYSE7</td> <td>SYMBOL LIST OVERFLOW</td></tr> <tr> <td>SYSE8</td> <td>ILLEGAL SECTION WORD</td></tr> <tr> <td>SYSE9</td> <td>ILLEGAL TAPE NAME</td></tr> <tr> <td>SYSE10</td> <td>TAPE NAME PREVIOUSLY DEFINED</td></tr> <tr> <td>SYSE11</td> <td>ILLEGAL ADDRESS PARAMETER</td></tr> </tbody> </table>	<u>Type-Out</u>	<u>Printout</u>	SYSE1	PROGRAM NOT ON SPECIFIED TAPE	*SYSE2	CHECKSUM ERROR	SYSE3	PROGRAM ORIGIN IN LIST AREA	SYSE4	PROGRAM ORIGIN IN COMMON AREA	SYSE5	UNDEFINED REFOUT(S)	SYSE6	ILLEGAL CARD	SYSE7	SYMBOL LIST OVERFLOW	SYSE8	ILLEGAL SECTION WORD	SYSE9	ILLEGAL TAPE NAME	SYSE10	TAPE NAME PREVIOUSLY DEFINED	SYSE11	ILLEGAL ADDRESS PARAMETER
<u>Type-Out</u>	<u>Printout</u>																								
SYSE1	PROGRAM NOT ON SPECIFIED TAPE																								
*SYSE2	CHECKSUM ERROR																								
SYSE3	PROGRAM ORIGIN IN LIST AREA																								
SYSE4	PROGRAM ORIGIN IN COMMON AREA																								
SYSE5	UNDEFINED REFOUT(S)																								
SYSE6	ILLEGAL CARD																								
SYSE7	SYMBOL LIST OVERFLOW																								
SYSE8	ILLEGAL SECTION WORD																								
SYSE9	ILLEGAL TAPE NAME																								
SYSE10	TAPE NAME PREVIOUSLY DEFINED																								
SYSE11	ILLEGAL ADDRESS PARAMETER																								

ERROR  
MESSAGES  
(Continued)

Type-Out	Printout
SYSE12	ILLEGAL PUN
SYSE13	TOO MANY PARAMETERS
SYSE14	PUN TABLE OVERFLOW
*SYSE15	JOB CARD INTERCEPTED
SYSE16	UNACCOUNTABLE MACHINE OR SYSTEM ERROR
*SYSE17	ILLEGAL SYS CONTROL INSTRU- TION
*SYSE18	ILLEGAL MNEMONIC
SYSE19	ILLEGAL DECIMAL NUMBER
*SYSE20	ILLEGAL POSITION FOR BINARY POINT
SYSE21	PARAMETER(S) MISSING
SYSE22	SYMBOL NOT IN SYMBOL LIST
*SYSE23	ILLEGAL FROM- TO PARAMETER
SYSE24	PARAMETER TOO LONG
SYSE25	COMMON SIZE EXCEEDS MEMORY
SYSE26	ILLEGAL BREAK CHARACTER
SYSE27	ILLEGAL MAGTAPE
SYSE28	NAME, FLAG TOO LONG
*SYSE29	ILLEGAL NUMBER OF WORDS TO BE REPLACED
SYSE30	ILLEGAL PARAMETER
SYSE31	TSYSTEM TSYSIN, TSYSLB CANNOT BE WRITE-ENABLED
*SYSE32	ILLEGAL PROG TAPE
SYSE33	ILLEGAL OCTAL NUMBER
SYSE34	TOO MANY BLOCKS TO BE READ
SYSE35	FORMAT FOR INTERNAL LOADER NOT SPEC.F.ED
SYSE36	READING TO SENTINEL WILL EXCEED MEMORY
*SYSE37	NO ADDRESS IN LOCATION FIELD
*SYSE38	ILLEGAL FORMAT
SYSE39	NEW COMMON BLOCK LENGTH EXCEEDS OLD
SYSE40	PROGRAM ORIGIN EQUALS ZERO
SYSE41	ILLEGAL USE OF TSYSIN IF SYS IN MAGTAPE MODE
SYSE42	ILLEGAL START-END PARAMETER
SYSE43	THIRD PARAMETER NOT (MASTER)
SYSE44	MASTER PROGRAM NOT DEFINED
SYSE45	SEGMENT AREA EMPTY

<u>ERROR MESSAGES (Continued)</u>	<u>Type-Out</u>	<u>Printout</u>
	SYSE46	A SYMBOL DEFINED MORE THAN ONCE WAS REFERENCED
	SYSE47	A REFERENCE SYMBOL WAS RE- DEFINED
	SYSE48	ILLEGAL TO LOCATE ON TSYSIN
	SYSE49	RELATIVE CSA EXCEEDS PMAX
	SYSE50	LIB MUST BE USED IF SUBS OR NOSUBS SPECIFIED
	SYSE51	ILLEGAL LIBRARY TAPE
	SYSE52	MORE THAN 7 LIBRARY TAPES SPECIFIED
	SYSE53	PARAMETER(S) SPECIFIED MORE THAN ONCE
	SYSE54	REFERENCE TO UNDEFINED COMMON BLOCK
	SYSE55	TAPE NAME TABLE OVERFLOW
	SYSE56	PARAMETER <u>nn</u> ILLEGAL
	SYSE57	PARAMETER <u>nn</u> MISSING
	SYS E58	Illegal CSA

## **IEXECUTI**

**IEXECUTI - Executes the Control Line in 1CONLIN + 1 to \*CONLIN + 9**

FUNCTION	Executes the control line in 1CONLIN+1 to 1CONLIN+9.
SYMBOLIC ADDRESS	SYS.IEXECUTI
PREDOMINANT USER	The programmer
PRESET REQUIREMENTS	None
ACTION	The 32KSYS control instruction in 1CONLIN +1 to 1CONLIN +9 is executed.  If an error occurs, exit is made to IXCONER.
EXIT	Control is returned to the next sequential instruction in the calling program.
REMARKS	See also IINTCON.

**1ZERO - Exponent Fault Exit**

FUNCTION	Initiates 32KSYS post-mortem dumping for a floating point exponent fault.				
SYMBOLIC ADDRESS	SYS.1ZERO				
PREDOMINANT USER	The programmer				
ACTION	Control is transferred to the 32KSYS post-mortem dump routine, the same as that used by 1ERRDMP. All actions listed for 1ERRDMP, except the type-out, apply to 1ZERO.				
REMARKS	<p>1. An exponent fault occurs whenever a floating point operation generates an exponent which cannot be entirely contained in the exponent field.</p> <p>2. If the programmer wishes, he may replace the automatic post-mortem dump entrance in 1ZERO with an entrance to his own routine. 1ZERO will be reset to the standard 32KSYS function by execution of a JOB control instruction.</p>				
CONSOLE TYPEWRITER TYPE-OUT	<table border="0"> <thead> <tr> <th style="text-align: center;"><u>Type-Out</u></th> <th style="text-align: center;"><u>Explanation</u></th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">ERRDMP0</td> <td style="text-align: center;">A programmed transfer of control to 1ZERO.</td> </tr> </tbody> </table>	<u>Type-Out</u>	<u>Explanation</u>	ERRDMP0	A programmed transfer of control to 1ZERO.
<u>Type-Out</u>	<u>Explanation</u>				
ERRDMP0	A programmed transfer of control to 1ZERO.				

## **SECTION XIV**

### **32KSYS SERVICE ROUTINES**

**32KSYS Service Routines** aid the programmer in performing routine functions of tape maintenance, debugging and program monitoring. They are located on TSYSTEM in RPL format and are called via their program identity as described below. Any or all of their particular functions are then performed depending upon the programmer's use of optional control instructions recognized by the routine itself. These control instructions, including the command, parameters, and break characters, are written in the same manner as that described for 32KSYS control lines.

#### **CALLING SERVICE ROUTINES**

Service routines may be called via their corresponding 32KSYS control lines (AIDE, TACSERV, RPLC, etc.). Execution of the instruction in the following example causes the service routine AIDE to be called into operation:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		AIDE	

## AIDE

### AIDE SERVICE ROUTINE

#### FUNCTION

Copies and/or compares information contained on magnetic tapes.

#### SERVICE ROUTINE INITIATION

L	Location	Command	Address and Remarks
		AIDE	

#### INPUT-OUTPUT SYSTEM

AIDE uses 32KSYS XORD for all tape operations.

#### CONTROL INSTRUCTIONS

AIDE accepts control instructions, written in standard 32KSYS control line format:

<u>Instruction</u>	<u>Function</u>
COPY	Copies information without change from one tape to another.
COMPF	Compares information on two tapes in a forward direction.
COMPB	Compares information on two tapes in a backward direction.
COPYCOMP	Copies information without change from one tape to another and then compares the information copied by reading both tapes in a backward direction.
SPACEF	Spaces forward over a tape.
SPACEB	Spaces backward over a tape.
LOCTACL	Locates a TACL (code mode) program.
LOCREL	Locates an REL program.

<u>Instruction</u>	<u>Function</u>
LOCRPL	Locates an RPL program.
LOCSENT	Locates a sentinel block.
WRTABS	Writes an ABS program.
WRTSENT	Writes a sentinel block.
REWIND	Rewinds one or more tapes.
ENDALL	Terminates the AIDE service routine.
LOCABS	Locates an ABS program.
REWINDLO	Rewinds with lockout one or more tapes.

The functions of REWIND, REWINDLO, LOCABS, LOCREL, LOCTACL, LOCRLPL, LOCSENT, WRTABS, WRTRPL, and WRTSENT are as described in Tape Handling.

**COPY** Copies information without change from one magnetic tape to another.

Format	L	Location	Command	Address and Remarks
			COPY	<u>t<sub>1</sub></u> , <u>t<sub>2</sub></u> , <u>n</u>
			COPY	<u>t<sub>1</sub></u> , <u>t<sub>2</sub></u> , <u>sent</u> , <u>c</u>

**Parameters**

- t<sub>1</sub> - The symbolic tape from which information is to be copied.
- t<sub>2</sub> - The symbolic tape to which information is to be copied.
- n - The number (decimal) of blocks.
- sent - The sentinel word to be used in locating a sentinel block.

## AIDE

c (OPTIONAL) - The mode of output to be used in indicating the number of blocks copied if sent is specified. If the parameter is omitted, T is assumed.

F Indicates the Console Typewriter.

T Indicates TSYSOUT, edited for the High-Speed Printer with Data Select 0.

B Both F and T.

### Action

If n is specified,

- n blocks of information are copied from t<sub>1</sub> to t<sub>2</sub>;
- t<sub>1</sub> and t<sub>2</sub> are positioned after the last block copied.

If sent is specified,

- information is copied from t<sub>1</sub> to t<sub>2</sub> until a sentinel block of sent is encountered;
- the sentinel block from t<sub>1</sub> is written twice on t<sub>2</sub> and t<sub>2</sub> is positioned between the sentinel blocks;
- t<sub>1</sub> is positioned after the sentinel block.

### Example

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		COPY	4, T-14, 512
		COPY	6, T-14, ZZZZZZZZ

Explanation: Execution of the first instruction causes 512 blocks of information from system program unit number 4 to be copied onto T-14. Execution of the second instruction causes the information from system program unit number 6 to be copied onto T-14 until a sentinel block of W/ZZZZZZZZ is detected.

**COMPF**

**Compares information on one magnetic tape with information on a second magnetic tape by reading both tapes in a forward direction.**

**Format**

L	Location	Command	Address and Remarks
		COMPF COMPF	<u>t<sub>1</sub></u> , <u>t<sub>2</sub></u> , <u>n</u> , <u>c</u> <u>t<sub>1</sub></u> , <u>t<sub>2</sub></u> , <u>sent</u> , <u>c</u>

**Parameters**

t<sub>1</sub> - The first symbolic tape to be compared.

t<sub>2</sub> - The symbolic tape to be compared with the first.

n - See COPY

sent - See COPY

c (OPTIONAL) - The mode of output to be used for indicating inequalities. If the parameter is omitted, T is assumed and only the result of the comparison (DIFFERENCES or NO DIFFERENCES) will be written on TSYSOUT.

F Indicates the Console Typewriter.

T Indicates TSYSOUT, edited for the High-Speed Printer with Data Select 0.

B Both F and T.

**Action**

- t<sub>1</sub> is compared with t<sub>2</sub> as specified by n or sent.
- If sent is specified, the sentinel blocks are also compared.
- If sent is specified, the comparison is terminated when a sentinel block of sent is encountered on t<sub>1</sub>.
- Inequalities are recorded on c.
- The tapes are positioned after the last block compared.

**AIDE**

<b>Example</b>	<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
			COMPF	4, T-14, 512, T
			COMPF	6, T-14, ZZZZZZZZ, F

**Explanation:** Execution of the first instruction causes 512 blocks of information on system program unit number 4 to be compared in a forward direction with 512 blocks of T-14. Execution of the second instruction causes the information on system program unit number 6 to be compared with T-14 until a sentinel block of W/ZZZZZZZZ is detected on 6.

**COMPB**

Compares information on one magnetic tape with information on a second magnetic tape by reading both tapes in a backward direction.

<b>Format</b>	<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
			COMPB COMPB	<u>t1, t2, n ,c</u> <u>t1, t2, sent, c</u>

**Remarks**

The parameters, action, and example for COMPB are the same as those of the COMPF control instruction, except that the comparison is made in a backward direction. If a begin-tape error occurs before the comparison is complete, the result of the comparison up to that point is written out according to c.

**COPYCOMP**

Copies information from one magnetic tape to another, and then compares the information copied by reading both tapes in a backward direction.

<b>Format</b>	<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
			COPYCOMP COPYCOMP	<u>t1, t2, n ,c</u> <u>t1, t2, sent, c</u>

**Parameters**

t<sub>1</sub> - The symbolic tape from which information is to be copied.

t<sub>2</sub> - The symbolic tape to which information is to be copied.

n - See COPY.

sent - See COPY.

c - See COMPF.

**Action**

- t<sub>1</sub> is copied to t<sub>2</sub>. (See Action of COPY.)
- t<sub>1</sub> is compared with t<sub>2</sub> in a backward direction. (See Action of COMPB.)
- The tapes are positioned as they were prior to the execution of the COPYCOMP function.

**Example**

	<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
			COPY	4, T-14, 512
			COPYCOMP	6, T-14, ZZZZZZZZ, F
			COMPB	4, T-14, 512, F

Explanation: Execution of the first instruction causes 512 blocks to be copied from system program unit number 4 onto T-14. Execution of the second instruction causes system program unit number 6 to be copied onto T-14 until a sentinel block of W/ZZZZZZZZ is detected on 6. Then 6 will be compared in a backward direction with T-14 and the number of blocks that were copied is indicated on the Console Typewriter. Execution of the third instruction causes 512 blocks on 4 to be compared in a backward direction with the same number of blocks on T-14. In both comparison cases, any inequalities will be typed on the Console Typewriter.

## AIDE

### SPACEF

Spaces forward over a magnetic tape.

#### Format

L	Location	Command	Address and Remarks
		SPACEF	<u>t, n</u>

#### Parameters

t - The symbolic tape to be spaced.

n - See COPY.

#### Action

t is spaced n blocks forward.

### SPACEB

Spaces backward over a magnetic tape.

#### Format

L	Location	Command	Address and Remarks
		SPACEB	<u>t, n</u>

#### Parameters

Same as SPACEF.

#### Action

t is spaced n blocks backward.

### ENDALL

Terminates action of the AIDE service routine.

#### Format

L	Location	Command	Address and Remarks
		ENDALL	

#### Action

The AIDE routine is terminated and control is returned to 32KSYS via 1NXTCON.

### ERRORS

#### Type-Out

#### Print-Out

AIDE E1	ILLEGAL CONTROL CARD
AIDE E2	PARAMETERS MISSING
AIDE E3	BEGIN-TAPE ERROR--NO SENTINEL OR TOO MANY BLOCKS
AIDE E4	MORE THAN 8 CHARACTERS IN SENTINEL PARAMETER

## AIDE

### Error Exit

Following detection and indication of an AIDE error, control is transferred to SYS.1ENDJOB.

### AIDE EXAMPLE

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		AIDE	\$ (1)
		REWIND ALPHA, GAMMA, BETA	\$ (2)
		COPY ALPHA, GAMMA, 150	\$ (3)
		COPYCOMP BETA, GAMMA, 335, F	\$ (4)
		COMPB ALPHA, GAMMA, 150, F	\$ (5)
		REWIND ALPHA, BETA	\$ (6)
		ENDALL	\$ (7)

Explanation: Execution of these instructions causes the following action to take place:

1. AIDE is called into operation.
2. ALPHA, BETA, and GAMMA are rewound.
3. 150 blocks of information are copied from ALPHA to GAMMA.
4. 335 blocks of information are copied from BETA onto GAMMA, and are then compared in a backward direction. Any differences are typed on the Console Typewriter.
5. 150 blocks of information on ALPHA and GAMMA (referred to in Step 3) are compared in a backward direction. Any differences are typed on the Console Typewriter.
6. Tapes ALPHA and BETA are rewound without lockout.
7. Control is returned to 32KSYS.

## ANALYZER

### ANALYZER SERVICE ROUTINE

#### FUNCTION

Prepares a listing of all references made by a program or a portion of a program to a specified area in memory. Any portion or all of memory may be specified.

#### SERVICE ROUTINE INITIATION

The ANALYZER service routine may be called into operation by the following instruction:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		ANALYZER	

#### PROGRAM TO BE ANALYZED

The program to be analyzed may be in RPL or ABS format and may appear on any magnetic tape except TSYSCR3, TSYOUT or TSYSCR6. It is not loaded into memory. An ABS program on tape must be preceded by a dummy PMAX card, unless the program to be analyzed immediately follows on TSYSIN.

#### INPUT-OUTPUT SYSTEM

ANALYZER uses 32KSYS XORD for all tape operations.

#### CONTROL INSTRUCTIONS

ANALYZER accepts two control instructions: ANALYZE and ENDALL. The instructions are written in standard 32KSYS control line format.

#### Format

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	<u>format</u>	ANALYZE	<u>t</u> , <u>id</u> , <u>addr1</u> , <u>addr2</u> , <u>addr3</u> , <u>addr4</u>

#### Parameters

format - Format of the program to be analyzed.

RPL      Indicates RPL format

ABS      Indicates ABS format

t - The symbolic tape except TSYSCR3, TSYOUT and TSYSCR6 that contains the input program.

id (OPTIONAL) - The identity of the program to be analyzed. If omitted, t will be assumed to be positioned at the start of the program.

## ANALYZER

addr1 - The beginning address of the coding to be analyzed.

addr2 - The ending address of the coding to be analyzed.

addr3 - The beginning address of the referenced area.

addr4 - The ending address of the referenced area.

### Action

- t is searched for program id.
- Each instruction of the program to be analyzed is then examined individually from tape to determine:
  1. If its location falls within the range addr1-addr2.
  2. If the memory address which it references falls within the range addr3-addr4.
  3. If the instruction contains a reference to an index register or is an index register-modified instruction.
- All instructions meeting conditions (1) and (2) or (1) and (3) are sequenced by referenced location.
- All processed output is then written onto TSYSOUT for printing on the High-Speed Printer in Data Select 0.
- t is normally positioned following the final block of the analyzed program.
- The next ANALYZER control instruction is requested.

### ANALYZER

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
----------	-----------------	----------------	----------------------------

EXAMPLE:

RPL	ANALYZE	ALPHA, ALTAC3, 10000, 22000.0, 77777\$
-----	---------	---

Explanation: Execution of this instruction causes a listing of RPL program ALTAC3, located on ALPHA to be prepared. The listing will include those references to memory locations 0 through 777778 made by program locations 100008 through 220008.

### OUTPUT FORMAT

The output of the ANALYZER service routine is written onto TSYSOUT for off-line printing on the High-Speed Printer with Data Select 0. Each page of output consists of a one-line heading statement, a listing of the sorted analyzed output, and an ending statement.

## **ANALYZER**

<b>Heading Statement</b>	The heading statement includes the date, time, page number and specified parameters.																		
<b>Ending Statement</b>	The ending statement is normally: END OF ANALYZER.																		
	If there was too much input data for the ANALYZER routine to process, the following ending statement will appear:																		
	<b>PROGRAM EXCEEDED ANALYZER MEMORY CAPACITY</b>																		
<b>REMARKS</b>	<ol style="list-style-type: none"><li>1. Processing of each ANALYZE instruction is completed before the next instruction is executed.</li><li>2. Separate outputs are written for each ANALYZE control instruction executed.</li><li>3. Programs need not be analyzed in the order they appear on tape since a complete search (forward and backward) is made for each program.</li></ol>																		
<b>ERRORS</b>	<table><thead><tr><th><u>Type-Out</u></th><th><u>Print-Out</u></th></tr></thead><tbody><tr><td>ANAL E1</td><td>ILLEGAL CONTROL CARD</td></tr><tr><td>ANAL E2</td><td>TAPE SPECIFIED IS RESERVED BY SYSTEM</td></tr><tr><td>ANAL E3</td><td>PARAMETER MISSING</td></tr><tr><td>ANAL E4</td><td>SPECIFIED FORMAT NOT ABS OR RPL</td></tr><tr><td>ANAL E5</td><td>ILLEGAL RPL SECTION WORD</td></tr><tr><td>ANAL E6</td><td>ILLEGAL ABS CARD</td></tr><tr><td>ANAL E7</td><td>CHECKSUM ERROR</td></tr><tr><td>ANAL E8</td><td>END PARAMETER LESS THAN START</td></tr></tbody></table>	<u>Type-Out</u>	<u>Print-Out</u>	ANAL E1	ILLEGAL CONTROL CARD	ANAL E2	TAPE SPECIFIED IS RESERVED BY SYSTEM	ANAL E3	PARAMETER MISSING	ANAL E4	SPECIFIED FORMAT NOT ABS OR RPL	ANAL E5	ILLEGAL RPL SECTION WORD	ANAL E6	ILLEGAL ABS CARD	ANAL E7	CHECKSUM ERROR	ANAL E8	END PARAMETER LESS THAN START
<u>Type-Out</u>	<u>Print-Out</u>																		
ANAL E1	ILLEGAL CONTROL CARD																		
ANAL E2	TAPE SPECIFIED IS RESERVED BY SYSTEM																		
ANAL E3	PARAMETER MISSING																		
ANAL E4	SPECIFIED FORMAT NOT ABS OR RPL																		
ANAL E5	ILLEGAL RPL SECTION WORD																		
ANAL E6	ILLEGAL ABS CARD																		
ANAL E7	CHECKSUM ERROR																		
ANAL E8	END PARAMETER LESS THAN START																		
<b>Error Exit</b>	Following detection and indication of the ANALYZER error, ANALYZER searches for the ENDALL card and then transfers control to 1NXTCON.																		

# DATA

## DATA SERVICE ROUTINE

### FUNCTION

Transfers data required by a user's program from TSYSSIN onto a specified tape. At the option of the user, the routine may also convert Image Mode input to Code Mode output and/or alter card format.

### SERVICE ROUTINE INITIATION

The DATA service routine may be called into operation by the following instruction:

L	Location	Command	Address and Remarks
		DATA	

### INPUT-OUTPUT SYSTEM

Data uses 32KSYS XORD for all tape operations.

### CONTROL INSTRUCTIONS

DATA accepts two control instructions: TAPE and ENDDATA.

The instructions are written in standard 32KSYS cor-ro-line format.

### TAPE

Specifies the data transfer to be performed by the DATA service routine, and must be immediately followed by the data to be transferred.

### Format

L	Location	Command	Address and Remarks
		TAPE	<u>t</u> , <u>mode</u> , <u>words</u> , <u>cards</u>

### Parameters

t - The symbolic tape except TSYSSOUT and TSYSTEM onto which the data is to be transferred.

mode (OPTIONAL) - The mode of the output data.

CODE  
or  
HOLL

Indicates Code Mode. Input may be either Code or Image Mode.

IMAGE  
or  
BINARY

Indicates Image Mode. Input must be Image Mode. If mode is omitted, input data will be transferred to t in Code Mode, 10 words per card 12 cards per block.

## DATA

words (OPTIONAL) - The number (decimal) of words per card to be transferred to the output tape. If mode is CODE or HOLL, words may be any number from one through 10. If mode is IMAGE or BINARY, words may be any number from one through 20.

cards (OPTIONAL) - The number (decimal) of cards per block to be transferred to the output tape.

The product of words times cards must be equal to or less than  $128_{10}$ .

### REMARKS

1. Cards is required if words is specified.
2. Blank columns on a card are transcribed onto t as zeros if the transfer is in Image Mode, or as spaces ( $60_8$ ) if the transfer is in Code Mode.
3. If the number of cards supplied as input data is not sufficient to meet the cards-per-block requirement for the final block, cards of all zeros or spaces will be provided for transfer to the output tape until the final block has been filled with the specified number of cards per block.
4. The remaining words in each block, determined by 128 minus words times cards will contain filler characters ( $32_8$ ).
5. If both words and cards are omitted, input data will be transferred to t as 10 words per card, 12 cards per block if mode is CODE or HOLL, 20 words per card six cards per block if mode is IMAGE or BINARY.
6. The TAPE control instruction must be the first card following the card for DATA.
7. t is not rewound by the DATA routine either before or after it is used.
8. Data is transcribed onto t starting at the beginning of the next block.

**ENDDATA**

Terminates the DATA service routine by writing the final output buffer block of data onto t, and causes control to be returned to 32KSYS.

**Format**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		ENDDATA	

**ERRORS**

<u>Type-Out</u>	<u>Print-Out</u>
DATA E1	ILLEGAL CONTROL CARD
DATA E2	TAPE SPECIFIED IS RESERVED BY SYSTEM
DATA E3	ILLEGAL FORMAT PARAMETER
DATA E4	INPUT IS NOT IMAGE AS SPECIFIED
DATA E5	WORDS/CARD NOT SPECIFIED OR ILLEGAL
DATA E6	CARDS/BLOCK NOT SPECIFIED OR ILLEGAL
DATA E7	WORDS/CARD X CARDS/BLOCK ILLEGAL
DATA E8	MORE THAN 8 CHARACTERS IN PARAMETER
DATA E9	PARAMETER IS NOT NUMERIC

**Error Exit**

Following the detection and indication of a DATA error, control is transferred to 1ENDJOB.

**DATA EXAMPLE**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		DATA	\$1
		TAPE	ALPHA, IMAGE, 16, 8 \$2
(Data to be transcribed)		ENDDATA	\$3
		DATA	\$4
		TAPE	BETA, CODE, 4, 30 \$5
(Data to be transcribed)		ENDDATA	\$6

Explanation: Execution of these instructions causes the following action to take place:

1. DATA is called into operation.

## DATA

2. ALPHA is specified as the tape onto which the Image Mode data is to be transferred in Image Mode, 16 words per card, 8 cards per block.
  - The data is processed and transferred as designated.
3. ENDDATA terminates the service routine.
4. The DATA instruction causes return of control to the DATA service routine.
5. BETA is the specified tape onto which the data is to be transferred. This transfer is to be in Code Mode, four words per card, 30 cards per block.
  - The data is processed and transferred as designated.
6. ENDDATA terminates the service routine. Control is returned to 32KSYS.

## FLID SERVICE ROUTINE

<b>FUNCTION</b>	<u>Finds or Lists the IDentities of RPL, ABS, REL, and TAC language programs on tape.</u>										
<b>SERVICE ROUTINE INITIATION</b>	The FLID service routine may be called into operation by the following instruction:										
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 2px;">L</th><th style="text-align: center; padding: 2px;">Location</th><th style="text-align: center; padding: 2px;">Command</th><th style="text-align: center; padding: 2px;">Address and Remarks</th></tr> </thead> <tbody> <tr> <td style="text-align: center; padding: 2px;"></td><td style="text-align: center; padding: 2px;"></td><td style="text-align: center; padding: 2px;">FLID</td><td style="text-align: center; padding: 2px;"></td></tr> </tbody> </table>			L	Location	Command	Address and Remarks			FLID	
L	Location	Command	Address and Remarks								
		FLID									
<b>INPUT REQUIREMENTS</b>	<p>Programs not starting at the beginning of a block will not be detected by FLID, and will be identified on the listing as OTHER. OTHER indicates that a block contains information not identifiable as RPL, ABS, REL, or TACL.</p> <p>ABS programs must be preceded by a dummy PMAX card.</p>										
<b>INPUT-OUTPUT SYSTEM</b>	FLID uses 32KSYS XORD for all tape operations.										
<b>CONTROL INSTRUCTIONS</b>	FLID accepts six control instructions, written in standard 32KSYS control line format:										
	<u>Instruction</u>	<u>Function</u>									
	REWIND	Rewinds one or more tapes.									
	LOCSENT	Locates a specified sentinel block.									
	SENTINEL	Designates an ending sentinel block for subsequent FIND or LIST control instructions.									
	FIND	Searches for a specified program.									
	LIST	Provides a listing of the type, ID, and block count of programs on a tape.									
	ENDALL	Terminates action of the FLID service routine and returns control to 32KSYS.									

## FLID

The functions of LOCSENT and REWIND are described in the TAPE HANDLING Section.

### SENTINEL

Presets FLID so that subsequent FIND or LIST control instructions will be terminated whenever the sentinel block indicated by this instruction is encountered.

#### Format

L	Location	Command	Address and Remarks
		SENTINEL	<u>sent</u>

#### Parameter

sent - The sentinel word to be used in locating a sentinel block.

#### Action

FLID is preset so that the operation of all subsequent FIND or LIST instructions - prior to another SENTINEL instruction - will be terminated whenever the sentinel block designated by sent is encountered.

#### Remarks

If the SENTINEL control instruction is omitted, a sentinel block of Z's is assumed for termination of FIND and LIST operations.

### FIND

Searches a tape for a program.

#### Format

L	Location	Command	Address and Remarks
		FIND	<u>t</u> , <u>format</u> , <u>id</u>

#### Parameters

t - The symbolic tape to be searched.

format - The type of program to be located.

RPL Indicates an RPL program.

ABS Indicates an ABS program.

REL Indicates a REL program.

TACL Indicates a TAC language program.

id - The ID of the program to be located.

- Action
- t is searched forward for program id.
  - When id is encountered, t is positioned at the beginning of the program.
  - If a sentinel block of Z's or the configuration specified by the SENTINEL control instruction is encountered, t is searched backwards. If the program is still not found, exit will be made to 1XCONER.

Example

	<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
			FIND	ALPHA, RPL, PROGRAMMA \$

Explanation: Execution of this instruction causes ALPHA to be searched for the RPL program PROGRAMMA. After the program is located, ALPHA is positioned at the beginning of PROGRAMMA. If a designated sentinel or Z block is encountered, the tape will be searched backward. If the program is still not found, exit will be made to 1XCONER.

LIST

Searches a tape and provides a listing of the type, ID, and block count of programs on the tape. The listing may be typed on the Console Typewriter and/or printed off-line on the High-Speed Printer.

Format

	<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
			LIST	<u>t, format, mode</u>

Parameters

t - The symbolic tape to be searched.

format - The type of programs to be listed.

RPL Indicates RPL programs.

ABS Indicates ABS programs.

REL Indicates REL programs.

## FLID

TACL Indicates TAC language programs.

ALL Indicates all of the above types of programs.

mode - The type of output for the listing.

T Indicates TSYSOUT for off-line printing  
on a High-Speed Printer.

B Indicates TSYSOUT for printing on the High-  
Speed Printer and the Console Typewriter.

F Indicates the Console Typewriter.

If the mode parameter is omitted, T is assumed.

### Action

- t is searched for the type of program(s) indicated by format.
- As each of the programs designated by format is encountered, its program type, ID and block count are typed on the Console Typewriter and/or are written on TSYSOUT, for off-line printing.
- The listing action is terminated whenever the designated sentinel or a block of Z's is encountered.

### Remarks

1. In addition to indicating the type, ID and block count of each of the programs designated by format, FLID lists all sentinels and a count of all blocks from the initial position on the tape, up to and including the final sentinel block.
2. If ALL is specified as format, blocks of information not recognized as RPL, ABS, REL, or TACL programs will be listed as ++++OTHER.
3. Data which appears in the same format as a sentinel block, i. e., contains 120 similar words, is listed as a sentinel but will not stop the listing function.
4. An RPL program which contains any unrecognizable RPL control word will be listed by FLID, but will be identified as a BAD RPL. The block containing the unrecognizable control word and any blocks remaining in the program will be identified as ++++OTHER.

5. If any sentinel block is encountered prior to an END card in a TACL program, or prior to a binary END or JMP card in an ABS or REL program, the listing will include:

- The program type as TACL or BINARY (the latter denoting an ABS or REL program),
- The first eight characters of the program iD,
- The words NO END (to denote omission of the TAC END card or the binary jump card), and
- The block count of the program.

<u>Example</u>	<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		LIST		ALPHA, TACL, B

**Explanation:** Execution of this instruction causes TACL programs and sentinel blocks on ALPHA to be listed until the designated sentinel or Z block is encountered. The listing plus a total block count of the tape is typed on the Console Typewriter and written on TSYSOUT.

**ENDALL** Terminates action and returns control to 32KSYS.

<u>Format</u>	<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
			ENDALL	

**Action** FLID is terminated and control is returned to 32KSYS.

<u>ERRORS</u>	<u>Type-Out</u>	<u>Print-Out</u>
FLID E1		ILLEGAL CONTROL CARD
FLID E2		ILLEGAL OR NO FORMAT SPECIFICATION
FLID E3		MORE THAN 8 CHARACTERS IN SENTINEL PARAMETER

## **FLID**

### **Error Exit**

**Following the detection and indication of a FLID error,  
FLID searches for the ENDALL card, and transfers  
control to 1NXTCON.**

**PLUM SERVICE ROUTINE****FUNCTION**

**Creates a new library or updates an old one.**

**SERVICE  
ROUTINE  
INITIATION**

The PLUM service routine may be called into operation by the following instruction:

L	Location	Command	Address and Remarks
		PLUM	

**INPUT-OUTPUT  
SYSTEM**

PLUM uses 32KSYS XORD for all reads, writes, and other tape movements.

**CONTROL  
INSTRUCTIONS**

PLUM accepts the following control instructions written in standard 32KSYS control line format:

<u>Instruction</u>	<u>Function</u>
OLDLIB	Specifies the tape and the identity of the library to be updated.
NEWLIB	Specifies the tape and the identity of the new library.
GENTAPE	Specifies the tape containing the generator to be added or replaced.
ADDGEN	Specifies the identity of the generator to be added or replaced.
ABSOLUTE	Indicates that absolute binary cards are to follow.
ENDGEN	Terminates the processing of a generator.
ADDMACRO	Specifies the name of the macro skeleton to be added or replaced.
ENDMACRO	Terminates the processing of a macro skeleton.

## PLUM

<u>Instruction</u>	<u>Function</u>
ADDSUB	Specifies the name of the TAC language subroutine to be added or replaced.
ENDSUB	Terminates the processing of a TAC language subroutine
ADDBRS	Specifies the name of a binary relocatable subroutine to be added or replaced
ME	Denotes that the subroutine (TAC language or binary relocatable) can also be entered at the symbolic location specified in the address field.
NEST	Specifies that the subroutine (TAC language or binary relocatable) to be added calls upon the subroutine specified in the address field.
DELETE	Deletes a macro generator, or subroutine from the library.
ENDALL	Terminates the PLUM routine and returns control to the SYSTEM via SYSIN/XICON
REWIND	Rewinds one or more magnetic tapes.

The function of REWIND is as described in the tape handling section

### POSITIONING ITEMS IN THE LIBRARY

When updating, routines must be prepared for addition, deletion or replacement in the input deck according to the order of the corresponding routines on the "old" library tape. The relative positioning of items to be added to the library is provided for by the "before" option and by the programmer's ordering of items in his input deck.

New generators and subroutines are added to the end of the new library tape in the order in which they appear unless the "before" option is used. Macros appear after the last macro of their specified generator unless the "before" option is exercised.

The "before" option is utilized by placing a semicolon after the name of the item to be added and following this semicolon with the name of the existing library item that the new item is to precede.

### Initialization Instructions

#### OLDLIB

Specifies the tape and the identity of the library being updated.

#### Format

L	Location	Command	Address and Remarks
		OLDLIB	<u>t</u> ; <u>oldid</u>

#### Parameters

t - The symbolic tape of the library being updated.

oldid - The identity of the library being updated.

#### Action

- If oldid and the identity of the library being updated are the same, PLUM requests the next control card which must be either REWIND or NEWLIB.
- If the identities are not the same, a console error type-out occurs indicating this condition. To continue the PLUM run, the correct library may be mounted and ADVANCE depressed.

#### Remarks

1. If \* is specified as t, TSYSLIB will be updated.
2. The OLDSLIB control instruction is omitted if a new library is being created (a library is not being updated).
3. An OLDSLIB control instruction may only be preceded by a REWIND instruction and may only be followed by:

REWIND  
NEWLIB

## PLUM

### NEWLIB

Specifies the tape and the identity of the new library.

#### Format

L	Location	Command	Address and Remarks
		NEWLIB	<u>t</u> <u>newid</u>

#### Parameters

t - The symbolic tape of the library being created.

newid - Eight or fewer alphanumeric characters that provide the identity of the library being created.

#### Action

- The identity of the new library is typed on the Console Typewriter and appears in the edited listing on TSYSOUT.
- PLUM is set to produce the library on t.

#### Remarks

1. Only one NEWLIB control instruction may be issued for each PLUM run.
2. If a library is being updated, the OLDDLIB instruction must precede the NEWLIB instruction.
3. Only the following control instructions are acceptable to PLUM at this time:

GENTAPE  
ADDSUB  
ADDMACRO  
ADDBRS  
ADDGEN  
DELETE  
REWIND  
ENDALL

## Updating Instructions

**GENTAPE**

Specifies the tape containing the generator to be added or replaced.

**Format**

L	Location	Command	Address and Remarks
		GENTAPE	<u>t</u>

**Parameter**

t - The symbolic tape containing the generator to be added to replaced.

**Remarks**

1. The ADDGEN control card must immediately follow the GENTAPE card or LIB E46 will result.
2. If all generators to be added or replaced during the PLUM run are on one tape, then only one GENTAPE control instruction need occur. However, each change in the generator tape must be indicated by a GENTAPE control instruction.

**ADDGEN**

Contains the identity of the generator to be added to the library tape.

**Format**

L	Location	Command	Address and Remarks
		ADDGEN	<u>n</u> ; <u>id</u>

**Parameters**

n - The number (decimal) of RPL blocks of the generator or an asterisk (\*).

id - The identity of the generator to be added to replaced.

**Action**

- If n is a decimal number, PLUM searches the generator tape (in a forward direction only) for the id specified and then copies that generator onto the library according to the block count.
- If n is an asterisk, the generator is added from absolute binary cards. The information from the cards is converted to RPL format and placed on the library tape. The id specified becomes the identity of the new RPL.

## PLUM

### Remarks

1. If n is an asterisk, the GENERATE control instruction is meaningless and need not occur.
2. If n is an asterisk, and less than 16 characters are specified as the id, the remaining characters of id (up to 16) are space filled.
3. The name and information cards which follow an ADDGEN card have the same meaning and format whether n is an asterisk or a block count.
4. If n is an asterisk, the ABSOLUTE control instruction must appear immediately following the last name and information card.
5. If n is a block count, the ENDDGEN control instruction must appear immediately following the last name and information card.
6. A macro generator differs from the normal generator in that a macro generator can have only one name.
7. A macro generator is added, replaced, or deleted in the same manner as a normal generator.
8. When a macro generator is replaced, the macros associated with it are not changed except as specified by subsequent control instructions.

### ABSOLUTE

Indicates that absolute binary cards are to follow.

#### Format

L	Location	Command	Address and Remarks
		ABSOLUTE	

#### Parameters

None.

#### Action

- Sets PLUM to convert the subsequent absolute binary cards to form the generator RPL.

#### Remarks

1. The control card is only required when the ADDGEN card contains an asterisk as its first parameter in the Address and Remarks field.
2. The next acceptable control card is ENDDGEN.

**ENDGEN**

Terminates the processing of a generator

**Format**

L	Location	Command	Address and Remarks
		ENDGEN	

**Parameters**

None.

**Action**

- Terminates generator-processing functions for current generator.
- Gets the next PLUM control instruction.

**Remarks**

Only the following control instructions are acceptable.

GENTAPE  
ADDMACRO  
ADDSUB  
ADDBRS  
ADDGEN  
DELETE  
REWIND  
ENDALL

**Example One:**

The following control cards will add a three name generator six blocks long to the library, from TSYSCR4:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>	<u>Explanation</u>
	GENTAPE	TSYSCR4		Symbolic Tape containing the generator.
	ADDGEN	6;GEN\$		Number of RPL blocks to be added; identity of generator is GEN.
		FIRSTGEN J;3		Multi-name of generator. Generator name produces a jump; sort # of 3.
		SECNAME N;79		Multi-name of generator. Generator name produces no jump; sort # of 79.
		THRNAME J;1		Multi-name of generator. Generator name produces a jump; sort # of 1.
	ENDGEN			Indicates end of name and information cards for GEN.

## PLUM

### Example Two:

The following control cards will add a generator from absolute binary cards before the routine, PLUMRTN:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>	<u>Explanation</u>
		ADDGEN	*-2RPLGEN	Generator is to be added from abs binary cards. Identity of generator is 2RPLGENΔ—Δ.
			GENABS; RTN	Name of generator to be added before RTN ("before" option)
			N 7	Generator name produces to jump sort # of 7. Indicates that absolute binary cards follow.
		ABSOLUTE	{ - - - - }	Binary cards to be converted to RPL. The last card must be a binary jump card.
		ENDGEN		Indicates the end of name and information cards for 2RPLGENΔ—Δ.

### ADDMACRO

Specifies the name of the macro to be added or replaced.

#### Format

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		ADDMACRO	<u>name</u>

#### Parameter

name - 8 or fewer alphanumeric characters that comprise the name of the macro

#### Action

The macro will be added at the end of the string of macros which follow the generator specified, unless the "before" option is used. If name already exists on oldid, a replacement occurs

#### Remarks

1. The macros immediately follow the macro generator with which they are associated
2. A macro generator may include 1023 macros.

**ENDMACRO**

Terminates the processing of a macro.

**Format**

L	Location	Command	Address and Remarks
		ENDMACRO	

**Parameters**

None.

**Action**

- Terminates macro-processing functions for current macro.
- Gets the next PLUM control instruction.

**Remarks**

Refer to remarks of ENDGEN card for acceptable control instructions.

**Example one:**

The control cards to add a macro would appear as follows:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>	<u>Explanation</u>
		ADDMARCO TWOMAC		Name of macro skeleton to be added
		MACGEN		Name of macro generator
		R;3		Must start on right; 3 half-words in length (after compilation).
		{ - - - - }		Macro skeleton coding
		ENDMACRO		

**Example Two:**

To add a macro using the "before" option, the control cards would appear as follows:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>	<u>Explanation</u>
		ADDMACRO ONEAMAC;TWOMAC		ONEAMAC to be added before TWOMAC
		MACGEN		Name of macro generator
		L;2		Must start on left; 2 half-words in length (after compilation).

## PLUM

### ADDSUB

Specifies the name of the TAC language subroutine to be added or replaced.

#### Format

L	Location	Command	Address and Remarks
		ADDSUB	<u>name</u>

#### Parameter

name - Eight or fewer alphanumeric characters that comprise the name of the TACL subroutine.

#### Action

The subroutine will be added at the end of the library unless the "before" option is used. If name already exists on oldid, a replacement occurs.

#### Remarks

1. Subroutines added with the ADDSUB control instruction must be terminated by an ENDSUB control instruction.
2. The language of the subroutine to be added is assumed to be TACL.
3. Binary relocatable routines may be included if preceded by, first, a BITSBITSBITSBITS card (see Example two, for format), and then by a binary relocatable PMAX card.
4. A TACLTACLTACLTACL card (see Example Two, for format) must be used to return to TACL mode.

### ENDSUB

Terminates the processing of a TAC language subroutine.

#### Format

L	Location	Command	Address and Remarks
		ENDSUB	

#### Parameters

None.

#### Action

- Terminates processing of the TACL subroutine.
- Gets the next PLUM control card

**Remarks**

1. The ENDSUB control instruction is recognized only when PLUM is in TACL mode (see remarks under ADDSUB).
2. Refer to remarks under ENDGEN for acceptable control cards at this time.

**ADDTRS**

Specifies the name of a binary relocatable subroutine to be added or replaced.

**Format**

L	Location	Command	Address and Remarks
		ADDTRS	<u>name</u>

**Parameter**

name - Eight or fewer alphanumeric characters that comprise the name of the REL subroutine.

**Action**

The indicated subroutine will be added at the end of the library, unless the "before" option is used. If name already exists on oldid, a replacement occurs.

**Remarks**

1. Subroutines added with ADDTRS control instruction are terminated by a binary relocatable END card.
2. The language of the subroutine to be added is assumed to be REL, and these routines must be preceded by a binary relocatable PMAX card.
3. TAC language routines may be included if preceded by a TACLTACLTACLTACL card (see Example Two, for format).
4. A BITSBITSBITS card must be used to return to REL mode.
5. Refer to remarks under ENDGEN for acceptable control cards at this time.

**ME**

Denotes that the subroutine (TACL or REL) can also be entered at the symbolic location specified in the address field.

## PLUM

Format	L	Location	Command	Address and Remarks
			ME	<u>symb</u>

Parameter            symb - Eight or fewer alphanumeric characters that comprise the multi-entrance of the subroutine.

NEST                Indicates that the TACL or REL subroutine to be added calls upon the subroutine specified in the address field.

Format	L	Location	Command	Address and Remarks
			NEST	<u>name</u>

Parameter            name - Eight or fewer alphanumeric characters that comprise the name of the nested subroutine.

Remarks             All ME cards must precede NEST cards.

Example One        To add a REL subroutine:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address &amp; Remarks</u>	<u>Explanation</u>
		ADDBRS	ALPHA	Name of subroutine
		ME	ABC	Denotes that ABC is another entrance to ALPHA
		NEST	DEF	Denotes that ALPHA calls on subroutine DEF
	(PMAX card)			Binary relocatable routine preceded by a PMAX card and terminated by an END card. When PLUM recognizes the binary relocatable END card, the action of the ADDBRS instruction is terminated.
		- - - - -		
		- - - - -		
		- - - - -		
		- - - - -		
		- - - - -		
		- - - - -		
		- - - - -		
	(REL-END card)			Terminates processing of subroutine.

### **Example Two**

To add a mixed TACL, REL, TACL subroutine:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>	<u>Explanation</u>								
	ADDSUB	BETA		Name of subroutine								
	ME	TUV		Multi entrance								
	NEST	WXY		Nested subroutine								
	NAME	BETA										
		- - - - -										
		- - - - -										
		- - - - -										
		- - - - -		TAC language cards								
		- - - - -										
B	ITSBITS	BITSBITS										
		- - - - -										
		- - - - -										
		- - - - -										
T	ACLTACL	TACLTACL		Binary relocatable cards preceded by a PMAX card								
		- - - - -										
		- - - - -										
		- - - - -		TAC language cards								
		ENDSUB										
DELETE			Deletes a generator, macro or subroutine from the library.									
Format				<table border="1"> <thead> <tr> <th>L</th><th>Location</th><th>Command</th><th>Address and Remarks</th></tr> </thead> <tbody> <tr> <td></td><td></td><td>DELETE</td><td><u>name</u></td></tr> </tbody> </table>	L	Location	Command	Address and Remarks			DELETE	<u>name</u>
L	Location	Command	Address and Remarks									
		DELETE	<u>name</u>									
Parameter				<u>name</u> - Eight or fewer alphanumeric characters specifying the name of the routine to be deleted.								
Action				<u>name</u> is deleted from the library.								
Remarks				<ol style="list-style-type: none"> <li>1. To delete a generator and its multiple names, from the library tape, all names must be deleted.</li> <li>2. Since a macro generator can have only one name, the generator and all its macros are deleted by specifying the name of the macro generator.</li> <li>3. Refer to remarks of ENDGEN card for acceptable control cards at this time.</li> </ol>								

## Wrapup Instruction

ENDALL

Terminates the PLUM routine and returns control to 32KSYS

# PLUM

Format	L	Location	Command	Address and Remarks
			ENDALL	

Parameters                          None

- Action
- An edited listing is produced on TSYSOUT for the High-Speed Printer.
  - A Console Typewriter typeout indicates the number of blocks of the new library (excluding the sentinel).
  - Tapes used by PLUM are rewound.
  - Return is made to 32KSYS via 1NXTCON.

CONSOLE TYPEWRITER NORMAL TYPE-OUTS	<u>Message</u>	<u>Explanation</u>	<u>Action</u>
	<u>NEWID</u> <u>newid</u> <u>nnnnn</u>	1Dof library being created. Number of blocks of new library.	None
CONSOLE TYPEWRITER ERROR TYPE-OUTS	<u>NOT LIB</u>	Tape to be updated is not a library tape.	Program halts at M/77777. Mount tape and depress ADVANCE. This can be used only once.
	<u>NOT ID</u> <u>oldid<sub>1</sub></u> <u>oldid<sub>2</sub></u>	The ID of the tape to be updated does not match that specified on the OLDLIB control card. <u>oldid<sub>1</sub></u> is the ID of the old library and <u>oldid<sub>2</sub></u> is the ID on the control card.	Program halts at M/77777. Mount new tape and depress ADVANCE. This option can be used only once.
	<u>PLUMDMP</u>	Computer error: dropped or picked up a bit ac- cording to the mask in the Q register.	Program halts at M/33333. Restart job.

CONSOLE  
TYPEWRITER  
ERROR  
TYPE-OUTS  
(Continued)

<u>Message</u>	<u>Explanation</u>	<u>Action</u>
LIB E <sub>n</sub>	PLUM recognized an illegal control card. <u>n</u> indicates the type of error and is explained below.	None required. Control is transferred to SYS.1ENDJOB.

ERROR  
PRINTOUTS

If PLUM detects an error, the card in error along with the LIB E<sub>n</sub> is printed on the system output tape, TSYS-OUT is edited for printing on the High-Speed Printer.

<u>Number</u>	<u>Explanation</u>
1	The initialization control cards are illegal.
2	While creating a new library, a DELETE control card was detected by PLUM.
3	While creating a new library, an ADDCARD with the "before" option was detected by PLUM.
4	The generator specified to be added to the library tape is not on tape specified in parameter of GENTAPE control.
5	OLDLIB is not a library tape. (1st word of 1st block was not ' LIBRARY. ')
6	A semicolon was missing from a control card, e.g., J, 3 on a generator control card which specifies a jump and sort number.
7	When operating on an ADDGEN "before" PLUM did not detect the routine which was specified by the "before" option.
8	The N or J, which specifies a jump or no jump on a generator control card, was missing.

## PLUM

### ERROR PRINTOUTS (Continued)

<u>Number</u>	<u>Explanation</u>
9	When updating an old library, the routine specified in a DELETE control card was not detected.
10	When operating on an ADDMACRO "before," PLUM did not detect the macro which was specified by the "before" option.
11	The name specified by the "before" option on an ADDMACRO control card was not a macro.
12	While creating a new library, a macro has been added out of the sequence of its macro generator.
13	While updating an old library, the macro generator for an ADDMACRO was not detected.
14	When operating on an ADDSUB "before", PLUM did not detect the routine which was specified by the "before" option.
15	When operating on an ADDBRS "before", PLUM did not detect the routine which was specified by the "before" option.
16	The ID of the library being updated does not agree with <u>id</u> of the OLDLIB control card.
17	A checksum error was detected on a binary relocatable card.
18	Legal control cards at this time are: ADDSUB, ADDBRS, ADDMACRO, ADDGEN, DELETE, REWIND, or ENDALL.
19	Working on ADDSUB. Legal control cards at this time are: ME, NEST, or NAME.
20	Working on ADDSUB. Legal control cards at this time are: NEST, or NAME.

**ERROR  
PRINTOUTS  
(Continued)**

<u>Number</u>	<u>Explanation</u>
21	Working on ADDSUB. Legal control cards at this time are: ENDSUB, or NAME.
22	Working on ADDBRS. Legal control cards at this time are: ME, NEST, or PMAX card.
23	Working on ADDBRS. Legal control cards are: NEST or PMAX card.
24	Working on ADDBRS. Legal control cards are: Binary relocatable end program card, ENDSUB, or NAME.
25	Working on ADDGEN. Legal card: One which specifies the name of a generator in the address field.
26	Working on ADDGEN. Legal control cards are: ENDGEN or a card containing the name of generator in address field.
27	Working on ADDGEN. Legal card: Jump or no Jump; sort number in address field.
28	Working on ADDMACRO: Legal card Name of generator, to which macro is to be added, in the address field.
29	Working on ADDMACRO. Legal card: Left right, or either; number of lines after compilation in the address field.
30	Working on ADDMACRO. Legal control card: ENDMACRO
31	The Table of Contents is too large.
32	Illegal binary relocatable subroutine card.
33	While adding a generator to an old library PLUM detected a subroutine or macro with the same name.

## PLUM

<u>Number</u>	<u>Explanation</u>
34	While adding a TACL or a binary relocatable to an old library, PLUM detected a macro or generator with the same name.
35	While adding a macro to an old library, PLUM detected that the generator specified was not a macro generator.
36	The "before" item specified in an ADDGEN, ADDSUB, or ADDBRS control card is a macro, or the second or later name of a multi-name generator.
37	The R, L, or E, which specifies right, left, or either on a macro input card, was missing.
38	Working on ADDGEN in absolute binary. Legal control cards at this time are: ABSOLUTE or spaces in Command Field.
39	Working on ADDGEN in absolute binary. PLUM encountered more than 75 names in a multi-name generator.
40	Working on ADDGEN in absolute binary. The only legal card immediately following an ABSOLUTE control card is an absolute binary card.
41	Working on ADDGEN in absolute binary. The only legal cards at this time are: an absolute binary card or a binary jump card.
42	Working on ADDGEN in absolute binary. The only legal cards immediately following a binary jump card are: ENDGEN or an absolute binary card.
43	A checksum error was detected on an absolute binary card. (This error may also indicate that the card is not an absolute binary card but appears to be one.)

**ERROR  
PRINTOUTS  
(Continued)**

<u>Number</u>	<u>Explanation</u>
44	GENTAPE parameter illegal.
46	Illegal control card following a GENTAPE card.
47	No GENTAPE command preceded an ADDGEN control card. Generator is added from a tape not used before.
48	Illegal parameter (minus sign) found on control card.

**Error Exit**

Following detection and indication of a PLUM error, control is transferred to SYS.1ENDJOB.

## RPLC

### RPLC SERVICE ROUTINE

#### FUNCTION

Revises the contents of a tape by adding, deleting, or correcting RPL programs.

#### SERVICE ROUTINE INITIATION

The RPLC service routine may be called into operation by the following instruction:

L	Location	Command	Address and Remarks
		RPLC	

#### INPUT-OUPUT SYSTEM

32KSYS XORD is used for all tape operations.

#### CONTROL INSTRUCTIONS

RPLC accepts the following written in standard 32KSYS control line format:

<u>Instruction</u>	<u>Function</u>
NEWTAPE	Specifies the tape onto which the updated information is to be written.
REWIND	Rewinds tapes.
COPY	Transfers information to the output tape specified by NEWTAPE.
COPYTIL	Transfers information until a designated RPL program is encountered.
SKIPTIL	Bypasses all information until a designated RPL program is encountered.
DELETE	Copies all information until a designated RPL program is encountered, and then bypasses the program.
ADD	Adds a program in RPL format.
WRTSENT	Writes a sentinel block on tape.

**CONTROL  
INSTRUCTIONS  
(Continued)**

<u>Instruction</u>	<u>Function</u>
LOCSENT	Locates a sentinel block on tape.
CORRECT	Makes corrections to an RPL, and transfers it to the output tape.
IDCHANGE	Copies all information on an input tape until a designated RPL program is encountered, makes corrections to that program and transfers it to the output tape. IDCHANGE also changes the RPL ID.
INDEX	Writes index block(s) on NEWTAPE that contain ID's of all programs on NEWTAPE to this point.
END	Terminates individual program correction.
ENDALL	Terminates the RPLC routine and returns control to SYS.

The functions of REWIND, LOCSENT, and WRTSENT are as described in TAPE HANDLING.

**NEWTAPE**

Specifies the magnetic tape onto which the updated information is to be placed.

L	Location	Command	Address and Remarks
		NEWTAPE	<u>t,</u>

**Format**

t - The symbolic tape (except TSYSTEM, TSYSOUT, and TSYSSIN) which is to be the output tape.

**Remarks**

NEWTAPE must precede all other control instructions except REWIND.

**COPY**

Copies all information from an input tape onto the output tape.

**Format**

L	Location	Command	Address and Remarks
		COPY	<u>t, sent</u>

## RPLC

### Parameters

t - The symbolic tape whose contents are to be copied.

sent (OPTIONAL) - The sentinel word to be used in locating a sentinel block.

### Action

- All information on t beginning at the point where t is currently positioned, is copied onto the output tape until a sentinel block of sent is encountered.
- If sent is omitted, a sentinel block of Z's will terminate the copy action.
- When the sentinel block is encountered, t is positioned immediately following the sentinel block.

### Example

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	COPY		ALPHA, AAAAAAAA

Explanation: Execution of this instruction causes all information on Tape ALPHA to be copied onto the output tape until a sentinel block of A's is encountered.

### COPYTIL

Copies information from an input tape onto the output tape until a specified RPL program is encountered.

### Format

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		COPYTIL	<u>t</u> , <u>id</u>

### Parameters

t - The symbolic tape whose contents are to be copied.

id (OPTIONAL) - The ID of the RPL program to be located. Omitting this parameter is equivalent to specifying the ID of the next RPL on the tape.

### Action

- All information on t, beginning where t is positioned, is copied onto the output tape until RPL program id is encountered.
- Upon completion of the copy, t is positioned at the start of the RPL program id.

### Remarks

The RPL Program id is not copied.

**SKIPTIL**

Bypasses all information on an input tape until a specified RPL program is encountered.

**Format**

L	Location	Command	Address and Remarks
		SKIPTIL	<u>t, id</u>

**Parameters**

Same as COPYTIL.

**Action**

- All information on t, from where t is positioned, is passed over until the RPL program id is encountered.
- When id is located, t is positioned at the beginning of that program.

**DELETE**

Copies all information from an input tape onto the output tape until a specified RPL program is located. That program is then bypassed.

**Format**

L	Location	Command	Address and Remarks
		DELETE	<u>t, id</u>

**Parameters**

Same as COPYTIL.

**Action**

- All information on t, from where t is currently positioned, is copied onto the output tape until the RPL program id is encountered.
- id is bypassed and t is positioned at the end of the program or immediately following the sentinel if one were encountered.

**ADD**

Bypasses all information on an input tape until a specified RPL program is located. That program is then copied onto the output tape.

**Format**

L	Location	Command	Address and Remarks
		ADD	<u>t, id</u>

## RPLC

### Parameters

t - The symbolic tape on which the RPL program is contained, or an asterisk (\*) which indicates that the ABS deck which follows is to be added to the output tape in RPL format.

id - The identity of the RPL to be added.

### Action

- If t is a symbolic tape, information on t, from where t is currently positioned, is bypassed until the RPL program id is encountered. The program is then copied to the output tape. t is positioned at the end of the program added or immediately following the sentinel if one were encountered.
  - If t is an asterisk (\*), the information from the ABS cards which immediately follow, is converted to RPL format and copied to the output tape.
1. If t is an \*, id is an optional parameter, if a PMAX card precedes the ABS deck.
  2. If both id and a PMAX card are present, the identity of the RPL is taken from the PMAX card.

### CORRECT

Copies all information from an input tape onto the output tape until a specified RPL program is located. Corrections are made to that program and it is copied onto the output tape.

### Format

L	Location	Command	Address and Remarks
		CORRECT	<u>t</u> , <u>id</u>

### Parameters

Same as ADD.

### Action

- If t is a symbolic tape, information on t from where t is currently positioned, is copied onto the output tape until the RPL program id is encountered.
- id is copied onto the output tape, and the corrections (which follow the CORRECT instruction) are added as separate RPL sections prior to the end of the RPL.
- t is positioned at the end of the program or immediately following a sentinel if one were encountered.

- If t is an asterisk (\*), see Action under ADD. In addition, the corrections (which follow the ABS END card) are added as separate RPL sections prior to the end of the RPL.

**Corrections**

Corrections to an RPL program appear immediately after the CORRECT or IDCHANGE control instructions, except where t in the CORRECT instruction is an asterisk (\*).

Format	L	Location	Command	Address and Remarks
	<u>p</u>	<u>m</u> 1	<u>instr</u>	<u>m</u> 2

**Parameters**

p - An alphabetic character, L or R, which indicates whether the change is to start in the left or right half of the word specified by m1. If omitted, L is assumed.

L	Specifies the left instruction
R	Specifies the right instruction

m1 - An octal address, written in the Location field, which indicates the location of the first word to be changed. If consecutive instructions or locations are to be corrected, the label and location of only the first need be specified.

instr - The new word constant (W/), octal constant (O/), or command part of a new instruction. Word constants may be eight or fewer alphanumeric characters. If fewer than eight characters are used, the constant is assumed to be left-justified and right-filled with spaces. Octal constants must contain exactly 16 digits. The command part of a new instruction is written as a standard TAC mnemonic.

m2 - An octal address. If indexed, the form must be: val,i where val is an octal number from 0-7777 and i is a number from 0-7.

**Remarks**

Only one correction per card is permitted.

## RPLC

**END** Terminates the individual program correction.

Format	L	Location	Command	Address and Remarks
			END	

**Parameters** None.

**Remarks** An END instruction must be used to terminate the correction procedure whether the t parameter of the CORRECT instruction is a symbolic tape or an asterisk (\*).

**IDCHANGE** Same as CORRECT. In addition, the identity of the corrected RPL is changed.

Format	L	Location	Command	Address and Remarks
			IDCHANGE	<u>t</u> , <u>id</u>

**Parameters** t - The symbolic tape containing the RPL program.

id - The identity of the RPL to be corrected.

- Action**
- Information on t, from where t is positioned, is copied onto the output tape until the RPL program id is encountered.
  - The Action for the CORRECT instruction applies, except that the ID of the corrected RPL program has been changed as specified by the first correction card.

**Remarks** The first correction following the IDCHANGE instruction must be the new ID. An I is written in the Label column and the new ID is written in the Command and Address and Remarks fields.

Example	L	Location	Command	Address and Remarks
	I		IDCHANGE PROGRAM2 END	ALPHA, PROGRAM1\$

**Explanation:** Execution of these instructions causes all information on Tape ALPHA to be copied onto the output tape until RPL program PROGRAM1 is encountered. When located, the old ID is replaced by PROGRAM2 and is copied onto the output tape. If there had been any other corrections, they would have been treated in the same manner as effected by the CORRECT control instruction. RPLC then requests its next control instruction.

**INDEX**

Writes an index block(s) onto the output tape that contains the ID's of all RPL programs already existing on NEWTAP

**Format**

L	Location	Command	Address and Remarks
		INDEX	

**PARAMETERS**

None.

**ACTION**

- The word \*\*\*INDEX is placed in Words 0 and 1 of the first index block. Succeeding blocks, if any, are not identified with \*\*\*INDEX.
- ID's of all RPL's on the output tape are transferred to the output block following the \*\*\*INDEX identification.
- When all ID's have been transferred, a sentinel word of 48/1's is placed in the output block following the last ID.

**REMARKS**

1. Index blocks that already exist on an input RPL tape are updated by RPLC when transferred to the output tape.
2. Index blocks can only be deleted by using RPLC control instructions to position the input tape around the index blocks.

**ENDALL**

Terminates the action of the RPLC service routine and returns control to 32KSYS.

**Format**

L	Location	Command	Address and Remarks
		ENDALL	<u>list</u> , <u>sent</u> , <u>pos</u>

## RPLC

### PARAMETERS

list (OPTIONAL) - Causes a list of the ID's of all RPL programs transferred to the output tape to be edited and written on TSYSOUT.

sent (OPTIONAL) - The sentinel word to be used in writing the sentinel blocks.

pos (OPTIONAL) - An alphabetic character that specifies the position of the output tape. If the parameter is omitted, the tape is positioned after the first sentinel block.

L      Specifies that the output tape is to be rewound with lockout.

R      Specifies that the output tape is to be rewound.

### ACTION

- Two sentinel blocks of Z's or the configuration specified by sent are written onto the output tape.
- If LIST is specified, a listing of the ID's of all RPL programs transferred to the output tape is edited and written on TSYSOUT. If list is omitted, no listing will be processed.
- The output tape is positioned according to pos.
- Control is returned to 32KSYS via INXTCON.

### RPLC SERVICE ROUTINE EXAMPLE

	<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		RPLC		(1)
		REWIND	NEWRPL, TAPE 10, TAPE 11, TAPE 12	(2)
		NEWTAPE	NEWRPL	(3)
		COPYTIL	TAPE 10, HOMER\$	(4)
		ADD	TAPE 11, ALPHA\$	(5)
		CORRECT	TAPE 12, GAMMA\$	(6)
L	14063	TMA	11075	(7)
R	15321	TDXL	4, 5	(8)
		END		(9)
		COPY	TAPE 10, AAAAAAAA	(10)
		ENDALL	LIST	(11)

**Explanation: Execution of these instructions causes the following action to take place:**

1. RPLC is called into operation
2. Tapes NEWRPL, TAPE 10, TAPE 11, and TAPE 12 are rewound.
3. Tape NEWRPL is specified as the final output tape.
4. All information on TAPE 10 is copied onto tape NEWRPL until RPL program HOMER is encountered. Tape remains positioned at the beginning of program HOMER.
5. All information on TAPE 11 is bypassed until RPL program ALPHA is encountered. ALPHA is copied onto NEWRPL. Tape 11 remains positioned at the beginning of the next block past ALPHA.
6. All information on TAPE 12 is copied onto NEWRPL until RPL program GAMMA is encountered.
7. GAMMA is copied onto NEWRPL and the TMA instruction is inserted as a separate RPL section at the end of GAMMA. At load time it will overlay location 14063L<sub>8</sub> of program GAMMA.
8. The TDXL instruction is inserted as a separate RPL section following the RPL section referred to in step 7. At load time, it will overlay location 15321R<sub>8</sub> of program GAMMA.
9. The END instruction signifies the end of corrections.
10. All information on TAPE 10 beginning with RPL program HOMER, is copied onto NEWRPL, until a sentinel block of A's is encountered.
11. Two sentinel blocks of Z's are written on NEWRPL, and a list of the ID's of all RPL programs transferred to the output tape is edited and written on TSY\$OUT. The output tape is positioned after the first sentinel block, and control is returned to 32KSYS via INXTCON.

## RPLC

ERRORS	<u>Type -Out</u>	<u>Print -Out</u>
	RPLC E1	ILLEGAL CONTROL CARD
	RPLC E2	TAPE SPECIFIED IS NEWTAPE OR IS A RESERVED SYSTEM TAPE
	RPLC E3	NEWTAPE HAS NOT BEEN SPECIFIED
	RPLC E4	SENTINEL BLOCK PREVENTED COM- PLETION OF FUNCTION
	RPLC E5	THE CARD FOLLOWING "IDCHANGE" CONTROL IS NOT AN "I" CARD
	RPLC E6	RPLC IS NOT GOOD - ILLEGAL SEC- TION WORD
	RPLC E7	ILLEGAL CORRECTION CARD (CARD IGNORED)
	RPLC E8	MORE THAN 8 CHARACTERS IN SENTINEL PARAMETER
	RPLC E9	ID IS MISSING ON ADDITION OR COR- RECTION OF AN ABS DECK
	RPLC E10	CHECKSUM ERROR ON ABS CARD
	RPLC E11	ILLEGAL BINARY CARD IN ABS DECK
	RPLC E12	ILLEGAL PARAMETER IN ENDALL CARD
Error Exit		Following detection and indication of an RPLC error, con- trol is transferred to SYS.1ENDJOB.

## TACSERV SERVICE ROUTINE

**FUNCTION**

Copies TAC language information from one magnetic tape to another, adding, deleting, or replacing cards or programs during the process.

**SERVICE  
ROUTINE  
INITIATION**

L	Location	Command	Address and Remarks
		TACSERV	

**INPUT  
REQUIREMENTS**

Information on the input tape must be in TAC language format in Code Mode, 10 words per card, 12 cards per block. Each program on the tape must contain an I card as the first card of the first block, and an END card as the final card of the program. An I card contains an I in Column 9, while Columns 10 through 16 are blank.

**INPUT-OUTPUT  
SYSTEM**

TACSERV uses 32KSYS XORD for all reads, writes, and tape movements.

**CONTROL  
INSTRUCTIONS**

TACSERV accepts 22 control instructions, written in standard 32KSYS control line format. The instructions are grouped into two classes - Class I and Class II - preceded by a mandatory initialization instruction, NEWTAPE. Class I instructions provide program-by-program and tape movement operations, while Class II instructions provide card-by-card operations for revision of TAC language programs. Class II is initiated by the CORRECT instruction and is usually terminated by the ENDPROG instruction.

The 22 TACSERV control instructions are:

<u>Instruction</u>	<u>Class</u>	<u>Function</u>
NEWTAPE	I	Initiation control card which assigns the output tape.
ADDPORG	I	Transfers a specified program without change from an input to the output tape.

## TACSERV

<u>Instruction</u>	<u>Class</u>	<u>Function</u>
COPY	I	Transfers information without change to the output tape until a specified sentinel block is encountered.
COPYTIL	I	Transfers information without change to the output tape until a specified program is encountered.
CORRECT	I	Transfers information without change to the output tape until a specified program is encountered, and sets TACSERV to the Class II mode (individual program corrections). Program correction instructions (Class II) must follow.
DELPROG	I	Deletes a specified program from an input tape.
ENDALL	I	Terminates the TACSERV routine and returns control to 32KSYS.
LOCATE	I	Searches for a specified program on an input tape.
LOCSENT	I	Searches for a specified sentinel block on an input tape.
REWIND	I	Rewinds one or more magnetic tapes.
REWINDLO	I	Rewinds with lockout one or more magnetic tapes.
SENTINEL	I	Designates an ending sentinel block.
WRTSENT	I	Writes a sentinel block on a specified tape.

The functions of REWIND, REWINDLO, LOCSENT, and WRTSENT are as described in TAPE HANDLING SECTION.

<u>Instruction</u>	<u>Class</u>	<u>Function</u>
ADD	II	Adds one or more cards to a program.
DELETE	II	Deletes one or more cards from a program.
DELADD	II	Deletes and adds cards.
ENDPROG	II	Terminates individual program corrections, and resets TACSERV to Class I mode.
LOCSEQ	II	Sets TACSERV to search for cards by sequence number.
LOCFLAD	II	Sets TACSERV to search for cards by symbolic address.
NOSEQ	II	Halts assignment of new sequence numbers within a program.
REPLACE	II	Replaces one or more cards in a program.
SEQ	II	Assigns new sequence numbers to cards within a program.

**NEWTAPE** Assigns the output tape.

Format	L	Location	Command	Address and Remarks
			NEWTAPE	<u>t</u>

**Parameter** t - The symbolic tape except system reserved tapes.  
which is to be the output tape.

## TACSERV

### Remarks

1. NEWTAPE must precede all other TACSERV instructions except REWIND.
2. t may not be used for any other TACSERV instruction except REWIND.

### CLASS I INSTRUCTIONS

Class I instructions perform all functions of the service routine except individual program corrections. TACSERV is preset to the Class I mode by the execution of the NEWTAPE control instruction. Only Class I instructions may be issued in this mode (individual program corrections).

### ADDPORG

Transfers a program item on input tape to the output tape.

#### Format

L	Location	Command	Address and Remarks
		ADDPORG	<u>t</u> <u>id</u>

#### Parameters

t - The symbol c tape from which a program is to be transferred.

id - The identity of the program to be added.

Note: An asterisk \* specified in place of t, id indicates that a TACL card deck follows the ADDPROG control card and is to be written onto the output tape.

#### Action

- All information on t item where t is positioned, is bypassed until the TACL program id is encountered.
- The program identified by id is copied onto the output tape and t is positioned at the beginning of the next block following that program.
- If ADDPROG \* is specified, all the cards following that control card, up to and including the first END card encountered, are transferred to the output tape.
- If the sentinel block is encountered prior to the program id, Error E4 will result.

**Example**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		ADDPORG	TSYSCRO, NAME 1\$

**Explanation:** Execution of this instruction causes all information on TSYSCRO beginning at the point at which the tape is currently positioned, to be bypassed until TACL program NAME 1 is encountered. Program NAME 1 is then copied onto the output tape and TSYSCRO is positioned at the start of the next block following program NAME 1.

**COPY**

Transfers without change all information from an input tape onto the output tape until a specified sentinel is located.

**Format**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		COPY	<u>t</u> , <u>sent</u>

**Parameters**

t - The symbolic tape to be copied.

sent (OPTIONAL) - The sentinel word to be used in locating a sentinel block. The sentinel specified by a COPY instruction overrides a previously-specified sentinel for the duration of that COPY instruction. If sent is omitted, Z's are assumed.

**Action**

- All information on t, from where t is positioned is copied onto the output tape until the sentinel block is encountered.
- The sentinel block is not copied onto the output tape and t is positioned at the start of the next block following the sentinel block.

**Example**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		COPY	T-024, AAAAAAAA

**Explanation:** Execution of this instruction causes all information on Tape T-024, beginning at the point at which the tape is currently positioned, to be copied onto the output tape until a sentinel block of A's is encountered.

**TACSERV****COPYTIL**

Transfers without change all information from a designated input tape to the output tape until a specified program is encountered.

**Format**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		COPYTIL	<u>t</u> <u>id</u>

**Parameters**

t - The symbolic tape to be copied.

id - The identity of the program to which the tape is to be copied.

**Action**

- All information on t, from where t is positioned, is copied onto the output tape until program id is encountered.
- t is positioned at the beginning of program id.
- If the sentinel block is encountered prior to program id Error E4 will result.

**Example**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		COPYTIL	T85.GAMMA \$

**Explanation** Execution of this instruction causes tape T85, beginning at the point at which it is currently positioned, to be copied onto the output tape until program GAMMA is encountered. The tape is positioned at the start of GAMMA.

**CORRECT**

Sets TACSERV to the Class II mode (individual program corrections).

**Format**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		CORRECT	<u>t</u> <u>id</u>

**Parameters**

t - The symbolic tape to be copied.

id (OPTIONAL) The identity of the program to be corrected. The id parameter may be omitted if the tape is pre-positioned.

- Action
- All information on t, from where t is positioned, is copied onto the output tape until program id is encountered.
  - t is positioned at the beginning of the designated program.
  - If id is omitted, corrections will be made from the point at which the input tape is currently positioned.
  - TACSERV is set to accept Class II instructions only.
  - TACSERV is set to search for cards in the LOCFLAD mode (by symbolic location). (Refer to LOCFLAD.)
  - If the specified sentinel block is encountered prior to program id Error E4 will result.

Remarks

The Class II mode is terminated by the execution of an ENDPORG control instruction.

Example	<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		CORRECT		OLDTACL, ZEBRA \$

Explanation: Execution of this instruction causes all information on Tape OLDTACL, beginning at the point at which the tape is currently positioned, to be copied onto the output tape until program ZEBRA is encountered. The tape is positioned at the start of program ZEBRA. Class II instructions must follow.

**DELPORG** Deletes a specified program.

Format	<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
			DELPORG	<u>t</u> , <u>id</u>

Parameters

t - The symbolic tape from which the program is to be deleted.

id - The identity of the program to be deleted.

## TACSERV

### Action

- t, from where t is positioned, is copied onto the output tape until program id is encountered.
- Program id is not copied onto the output tape, but is bypassed and the tape is positioned at the beginning of the block following the program or immediately following the sentinel block, if one were encountered.
- If the sentinel block is encountered prior to the program designated by id Error E4 will result.

### Example

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		DELPROG	T72, YOKE \$

Explanation: Execution of this instruction causes all information on Tape T72, beginning at the point at which the tape is currently positioned, to be copied onto the output tape until program YOKE is encountered. Program YOKE is not copied onto the output tape, but is bypassed and the input tape is positioned at the start of the next block following the program.

### ENDALL

Terminates the TACSERV service routine.

### Format

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		ENDALL	<u>list</u> , <u>sent</u> , <u>pos</u>

### Parameters

- list (OPTIONAL) - Produces a listing on TSYOUT of the ID and block count of each program on the output Tape, plus the total block count including sentinels.
- sent (OPTIONAL) - The sentinel word to be used in writing the sentinel blocks. If sent is omitted, Z's are assumed.
- pos (OPTIONAL) - An alphabetic character that specifies the position of the output tape. If the parameter is omitted, the tape is positioned after the first sentinel block.
- L Specifies that the output tape is to be rewound with lockout.
- R Specifies that the output tape is to be rewound.

## TACSERV

### Action

- Two sentinel blocks of Z's or the configuration specified by sent are written onto the output tape.
- If LIST is specified, a listing of the ID and block count of each program on the output tape, plus the total block count including sentinels, will be produced for printing on the High-Speed Printer. If LIST is omitted, no listing will be processed.
- The output tape is positioned according to the pos parameter.
- Action of the TACSERV routine is terminated and control is returned to 32KSYS via 1NXTCON.

### LOCATE

Search for a designated program on an input tape.

### Format

L	Location	Command	Address and Remarks
		LOCATE	<u>t</u> , <u>id</u>

### Parameters

t - The symbolic tape on which the designated program is located.

id - The identity of the program to be located.

### Action

- t is searched for program id.
- When program id is encountered, the tape is positioned at the start of the program.
- If the sentinel block is encountered prior to program id, Error E4 will result.

### Example

L	Location	Command	Address and Remarks
		LOCATE	T-024, XRAY \$

Explanation: Execution of this instruction causes Tape T-024 to be searched for program XRAY and the tape to be positioned at the start of that program.

## TACSERV

### SENTINEL

Designates an ending sentinel block.

#### Format

L	Location	Command	Address and Remarks
		SENTINEL	<u>sent</u>

#### Parameter

sent - The sentinel word to be used in locating the sentinel block.

#### Action

- There is no immediate action. TACSERV is set so that the operation of all subsequent instructions - prior to another SENTINEL instruction - will be terminated whenever a sentinel block of sent is encountered.

#### Remarks

If the SENTINEL control instruction is omitted, a sentinel block of 120 words of Z's will be assumed by TACSERV for termination of subsequent instruction action.

### CLASS II INSTRUCTIONS

Class II instructions perform individual program corrections such as the addition, deletion, or replacement of cards. TACSERV is set to accept Class II instructions by the execution of the CORRECT instruction. Only Class II instructions may be issued in this mode; otherwise, Error E1 will result.

Only one program may be processed for each CORRECT instruction. Corrections to a program must be made in the sequential order of the program cards.

This mode is terminated by the processing of an ENDPROG card.

### TRANSFER OF DATA

Whenever an ADD, REPLACE, DELADD, or DELETE instruction is executed, information is transferred either from the beginning of the program being corrected, or from the point of last change in that program (via a previous ADD, REPLACE, DELADD, or DELETE instruction) to the point at which cards are to be currently added, replaced, or deleted. Whenever the ENDPROG instruction is executed, the balance of the program is transferred from the input tape to the output tape, and the Class II phase is terminated.

**ADD**

Adds one or more cards to a program.

**Format**  
**LOCFLAD**  
**mode**

L	Location	Command	Address and Remarks
		ADD	<u>symb, n, total</u>

**Parameters**  
**LOCFLAD**  
**mode**

symb - The symbolic location used with n that designates the card in front of which the new card(s) are to be added.

n - A decimal number that indicates the number of cards beyond symb, ahead of which the cards are to be added. If n is zero, the card identified by symb is the one ahead of which the new cards will be added.

total (OPTIONAL) - The number (decimal) of consecutive cards to be added. If total is omitted, cards will be added until a TACSERV Class II instruction is encountered.

**Format**  
**LOCSEQ**  
**mode**

L	Location	Command	Address and Remarks
		ADD	<u>xxxxxxxx, total</u>

**Parameters**  
**LOCSEQ**  
**mode**

xxxxxxxx - The eight-character identity and Sequence field of the card in front of which new cards are to be added.

total (OPTIONAL) - See description under LOCFLAD mode.

**Action**

- The TACL program specified by the most recent CORRECT control instruction is copied card-by-card onto the output tape until the specified card is encountered. The specified card is not transferred to the output tape and remains available for further processing.
- The cards to be added are written onto the output tape. The next TACSERV control instruction is then requested.

## TACSERV

### Remarks

1. Cards are always added preceding the card specified by symb + n in LOCFLAD mode or xxxxxxxx in LOCSEQ mode.
2. If symb is blank or zero, cards will be added immediately preceding the nth card beyond the one last affected by an ADD, DELADD, DELETE, or REPLACE instruction for this program. If there were no such previous instructions, the cards will be added prior to the nth card from the beginning of the program.
3. If symb and n are both zero, the current location on the input tape is assumed.
4. The cards to be added must immediately follow the ADD control instruction.
5. If an END card is added, it must be followed by an ENDPORG card. Cards between the END card being added and the ENDPORG card will be ignored.

### ADD Example LOCFLAD mode

Assume that two instructions are to be added in LOCFLAD mode, between the third and fourth instructions of the following coding:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	TEST	CA	
	SLAQ		6
	TMD		W/00000001
	JMP		START

The following TACSERV instructions are used to add the new cards:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		ADD	TEST, 3, 2
		JAED	(P) + 2H
		AM	ZEBRA

Execution of the TACSERV instruction ADD causes the preceding portion of the program, from either its beginning or point of last change to the JMP START card (the card indicated by TEST +3) to be written onto the output tape. At this point, the two cards following ADD are written following the TMD W/00000001 card on the output tape. The input tape remains positioned at the JMP START card.

The corrected portion of the program on the output tape now appears as

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	TEST	CA	
	SLAQ		6
	TMD		W/00000001
	JAED		(P) +2H
	AM		ZEBRA
	JMP		START

**ADD Example  
LOCSEQ mode**

Assume that two instructions are to be added in LOCSEQ mode preceding the fourth instruction of the coding below.

<u>Identity and Sequence</u>	<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
XRAY0012			CA	
XRAY0013			SLAQ	6
XRAY0014			TMD	W/00000001
XRAY0015			JMP	START

The following TACSERV instructions are used to enter the new cards:

<u>Identity and Sequence</u>	<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
			ADD	XRAY0015
			JAED	(P) + 2H
			AM	ZEBRA
			ENDPROG	

Execution of the TACSERV instruction ADD causes the preceding portion of the program, from either its beginning or point of last change up to, but not including,

## TACSERV

card XRAY0015 (JMP START), to be written onto the output tape. At this point, the two cards following ADD are written following the TMD W/00000001 card onto the output. ENDPORG indicates to TACSERV that there are no further corrections to be made to this program, and the remainder of the program is transferred to the output tape.

The corrected portion of the program on the output tape now appears as:

<u>Identity and Sequence</u>	<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
XRAY0012			CA	
XRAY0013			SLAQ	6
XRAY0014			TMD	W/00000001
			JAED	(P) + 2H
			AM	ZEBRA
XRAY0015			JMP	START

### DELADD

Deletes one or more cards from a TACL program, then adds cards until the next TACSERV control is encountered.

Format  
LOCFLAD  
mode

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		DELADD	<u>symb, n, total</u>

Parameters  
LOCFLAD  
mode

symb - The symbolic location used in conjunction with n that designates the first card to be deleted.

n - The number (decimal) of cards past the card (designated by symb) of the first card to be deleted. If n is zero, the card identified by symb will be the first to be deleted.

total (OPTIONAL) - The number (decimal) of cards to be deleted. If total is omitted, one card will be deleted.

Format  
LOCSEQ  
mode

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		DELADD	<u>XXXXXXXX, total</u>

<b>Parameters</b>	<u>XXXXXXXX</u> - The eight-character Identity and Sequence field of the first card to be deleted.																																				
LOCSEQ mode	<u>total</u> (OPTIONAL) - The number (decimal) of cards to be deleted. If <u>total</u> is omitted, one card will be deleted.																																				
Action	<ul style="list-style-type: none"> <li>• The TACL program designated by the most recent CORRECT instruction is copied card-by-card onto the output tape until the specified card is encountered.</li> <li>• Cards are transferred from TSYSIN to the output tape until a TACSERV Class II instruction is encountered. Then the <u>total</u> number of cards starting with the specified card is deleted.</li> </ul>																																				
Remarks	<ol style="list-style-type: none"> <li>1. If <u>symb</u> is blank or zero, cards will be deleted from the <u>n</u>th card beyond the last one referenced in a previous ADD, DELADD, DELETE or REPLACE instruction for this program. If there were no such previous instructions, cards will be deleted from the <u>n</u>th card of the program.</li> <li>2. If <u>symb</u> and <u>n</u> are both zero, the current location on the input tape is assumed.</li> <li>3. The processing of an END card will terminate the DELADD function. If the END card is detected during the delete phase, Error E11 will result.</li> </ol>																																				
DELADD Example LOCFLAD mode	Assume that the third and fourth instructions are to be deleted in LOCFLAD mode from the following coding, and three instructions are to be added.																																				
	<table border="1"> <thead> <tr> <th><u>L</u></th> <th><u>Location</u></th> <th><u>Command</u></th> <th><u>Address and Remarks</u></th> </tr> </thead> <tbody> <tr> <td></td><td>TEST</td><td>CA</td><td></td></tr> <tr> <td></td><td>SLAQ</td><td></td><td>6</td></tr> <tr> <td></td><td>TQD</td><td></td><td></td></tr> <tr> <td></td><td>JMP</td><td>OUT</td><td></td></tr> <tr> <td></td><td>TMD</td><td>W/00000001</td><td></td></tr> <tr> <td></td><td>JAED</td><td>(P) + 2H</td><td></td></tr> <tr> <td></td><td>AM</td><td>ZEBRA</td><td></td></tr> <tr> <td></td><td>JMP</td><td>START</td><td></td></tr> </tbody> </table>	<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>		TEST	CA			SLAQ		6		TQD				JMP	OUT			TMD	W/00000001			JAED	(P) + 2H			AM	ZEBRA			JMP	START	
<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>																																		
	TEST	CA																																			
	SLAQ		6																																		
	TQD																																				
	JMP	OUT																																			
	TMD	W/00000001																																			
	JAED	(P) + 2H																																			
	AM	ZEBRA																																			
	JMP	START																																			

## TACSERV

The following TACSERV instruction is used to delete the two instructions and add three in their place.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		DELADD	TEST 2,2
		TQM	ERR
		TMQ	MASKI
		JMP	OUTA

Execution of the TACSERV instruction DELADD causes the preceding portion of the program, from either its beginning or point of last change, up to but not including the TQD instruction, to be written on the output tape. At this point, the TQD instruction and the one immediately following it are bypassed on the input tape and the tape is positioned at the TMD W/00000001 card.

The corrected portion of the program on the output tape now appears as:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		TEST	CA
		SLAQ	6
		TQM	ERR
		IMO	MASKI
		JMP	OUTA

The card TMD W/00000001 and those following remain on the input tape and are available for further processing.

### DELETE

Deletes one or more cards from a TACL program.

Format  
LOCFLAD  
mode

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		DELETE	<u>symb</u> <u>n</u> , <u>total</u>

Parameters  
LOCFLAD  
mode

symb - The symbolic location used in conjunction with n that designates the first card to be deleted.

n - The number (decimal) of cards past the card (designated by symb) of the first card to be deleted. If n is zero, the card identified by symb will be the first to be deleted.

total (OPTIONAL) - The number (decimal) of cards to be deleted, or TO END to delete all cards from where the program is positioned, up to but not including the END card of the program being revised. If total is omitted, one card will be deleted.

Format  
LOCSEQ mode

L	Location	Command	Address and Remarks
		DELETE	<u>xxxxxxxx, total</u>

Parameters  
LOCSEQ mode

xxxxxxxx - The eight-character Identity and Sequence field of the first card to be deleted.

Action

The TACL program designated by the most recent CORRECT instruction is copied card-by-card onto the output tape until the specified card is encountered.

The specified card (first card to be deleted) plus all subsequent consecutive cards as determined by total are bypassed on the input tape. The input tape is positioned at the card following the last one bypassed.

If TO END is specified as total, all cards of the program being revised, from the specified card up to but not including the END card, will be bypassed (deleted). The input tape will then be positioned at the END card.

Remarks

1. If symb is blank or zero, cards will be deleted starting with the nth card beyond the last one referenced in a previous ADD, DELETE or REPLACE instruction for this program. If there were no such previous instructions, cards will be deleted beginning with the nth card of the program.
2. If symb and n are both zero, the current location on the input tape is assumed.
3. An END card of a program being revised cannot be deleted.

## TACSERV

### DELETE Example LOCFLAD mode

Assume that the third and fourth instructions are to be deleted in LOCFLAD mode from the following coding.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	TEST	CA	
		SLAQ	6
		TQD	
		JMP	OUT
		TMD	W/00000001
		JAED	(P) + 2H
		AM	ZEBRA
		JMP	START

The following TACSERV instruction is used to delete the two instructions.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		DELETE	TEST, 2, 2

Execution of the TACSERV instruction DELETE causes the preceding portion of the program, from either its beginning or point of last change, up to but not including the TQD instruction, to be written on the output tape. At this point, the TQD instruction and the one immediately following it are bypassed on the input tape and the tape is positioned at the TMD W/00000001 card.

The corrected portion of the program on the output tape now appears as:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	TEST	CA	
		SLAQ	6

The card TMD W/00000001 and those following remain on the input tape and are available for further processing.

### ENDPROG

Terminates individual program correction.

### Format

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		ENDPROG	

**Parameters**

None

**Action**

- The balance of the program being corrected, from the point at which the tape is currently positioned, is copied onto the output tape.
- The input tape is positioned at the beginning of the block following the last block of the program just processed.
- TACSERV is set to NOSEQ mode, and reset to accept Class 1 instruction.

**Remarks**

An ENDPORG card must always be used to terminate the correction of a program.

**LOCFLAD**

Sets the TACSERV routine to find cards by symbolic address for subsequent ADD, DELADD, DELETE, and REPLACE control instructions.

**Format**

L	Location	Command	Address and Remarks
		LOCFLAD	

**Parameters**

None

**Action**

- There is no immediate action. All subsequent card searching within the current CORRECT instruction phase is set to use symbolic addresses in the Location field.

**Remarks**

1. TACSERV is automatically set to the LOCFLAD mode of operation whenever a CORRECT instruction is executed.
2. The LOCFLAD mode is terminated by the execution of a LOCSEQ instruction.

**LOCSEQ**

Sets the TACSERV routine to find cards by Identity and Sequence number for subsequent ADD, DELADD, DELETE, and REPLACE control instructions.

**Format**

L	Location	Command	Address and Remarks
		LOCSEQ	RIGHT

## \* TACSERV

### Parameters

**RIGHT-(OPTIONAL)** - Specifies that the sequencing and locating of cards by identity and sequence numbers will be done on the right-hand side of cards, columns 73 thru 80. If RIGHT is omitted, card searching and sequencing will be performed on the left-hand side of the cards, columns 1 through 8.

### Action

There is no immediate action. All subsequent card searching within the current CORRECT instruction chase will be performed according to the above parameter specifications.

### REMARKS

1. If RIGHT is specified subsequent LOCFLAD, LOCSEQ, or SEQ instructions will not terminate right-hand side sequencing. Right-hand sequencing can only be terminated by the execution of NOSEQ or ENDPORG instruction.
2. The LOCSEQ mode of card-searching is terminated by the execution of a LOCFLAD or ENDPORG instruction.

### NOSEQ

Terminates assignment of new sequence numbers to cards of a TACL program being corrected. (Assignment of sequence numbers is initiated by the SEQ control instruction).

### Format

L	Location	Command	Address and Remarks
		NOSEQ	

### Parameters

None

### Action

- There is no immediate action. When NOSEQ is executed, all cards written on the output tape from this point forward will be transferred with their Identity and Sequence field unaltered.

### REPLACE

Replaces one or more cards in the TACL program. REPLACE operates either in LOCFLAD or LOCSEQ mode.

### Format LOCFLAD mode

L	Location	Command	Address and Remarks
		REPLACE	<u>symb</u> , <u>n</u> , <u>total</u>

### Parameters LOCFLAD mode

symb - The symbolic location used in conjunction with n of the first card to be replaced.

n - The number (decimal) of cards past the card (designated by symb) of the first card to be replaced. If n is zero, the card identified by symb will be the first to be replaced.

total (OPTIONAL) - The number (decimal) of consecutive cards to be replaced. If total is omitted, one card will be replaced.

**Format  
LOCSEQ  
mode**

L	Location	Command	Address and Remarks
		REPLACE	<u>xxxxxxx</u> , <u>total</u>

**Parameters  
LOCSEQ mode**

xxxxxxx - The eight-character Identity and Sequence field of the first card to be replaced.

total (OPTIONAL) - The number (decimal) of consecutive cards to be replaced. If total is omitted, one card will be replaced.

**Action**

- The TACL program specified by the most recent CORRECT instruction is copied card-by-card onto the output tape until the specified card is encountered.
- The specified card plus subsequent cards as determined by total are replaced. Their corresponding replacement cards (following the REPLACE instruction) are written onto the output tape. The input tape is positioned at the card following the last one replaced.

**Remarks**

1. If symb is blank or zero, cards will be replaced starting with the nth card beyond the last one referenced in a previous ADD, DELETE, or REPLACE instruction for this program. If there were no previous such instructions, cards will be replaced beginning with the nth card of the program.
2. If symb and r are both zero, the current location on the input tape is assumed.
3. The REPLACE function will be terminated whenever an END card is processed. An END card may be used to replace any other card, but an END card must not be replaced by another card.

**REPLACE Example  
LOCFLAD mode**

Assume that the third and fourth instructions are to be replaced in LOCFLAD mode in the following coding.

L	Location	Command	Address and Remarks
		TMD	W/00000001
		JAED	(P) + 2H
	TEST	CA	

## TACSERV

<b>REPLACE</b>	<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
<b>Example</b>				
<b>LOCFLAD Mode</b>			SLAQ	6
<b>(Continued)</b>			TMQ	W/77777777
			JAEQ	(P) + 3H
			AM	ZEBRA
			JMP	START

The following TACSERV instruction is used to replace the two instructions.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	REPLACE	TEST, 2.2	

Execution of the TACSERV instruction REPLACE causes the preceding portion of the program, from either its beginning or point of last change up to but not including, the TMQ W/77777777 card (indicated by TEST+2), to be written on the output tape. At this point, the TMQ W/77777777 card and the one following on the input tape are bypassed. The two cards following REPLACE are written on the output tape following SLAQ 6.

The input tape remains positioned at the AM ZEBRA card, which is available for further processing.

The revised portion of the program on the output tape now appears as:

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	TEST	CA	
		SLAQ	6
		TMD	W/00000001
		JAED	(P) + 2H

### SEQ

Sets TACSERV to assign new sequence numbers, beginning with a designated configuration, to each subsequent card of a TAC language program being transferred to the output tape.

**Format**

L	Location	Command	Address and Remarks
		SEQ	<u>seqno, inc, cols</u>

**Parameters**

seqno (OPTIONAL) - Eight alphanumeric characters to be placed in the Identity and Sequence field of the next output card of the program being revised. The seqno parameter may consist of zero to eight alphabetic characters or spaces followed by zero to eight decimal characters. The total must be eight, representing the eight columns of the Identity and Sequence field. (If seqno is omitted, the Identity and Sequence field of the first card transferred to the output tape will be transferred as the first sequence number.) See Note: below

inc (OPTIONAL) - The number (decimal) by which seqno is to be incremented from card to card. If inc is omitted, an increment of one will be assumed.

cols (OPTIONAL) - The number (1 through 8) of columns of seqno that will be affected by the incrementing process. If cols is omitted, four columns will be assumed.

**Action**

- There is no immediate action. All cards transferred to the output tape following execution of the SEQ instruction will be sequenced according to the above parameter specifications.

**Remarks**

1. Incrementation numbering is modulo  $10^c$ , where c is the number of columns to be affected.
2. Sequencing is terminated by the NOSEQ control instruction or the ENDPORG control instruction.
3. Each of the SEQ parameters may be used independently of the others, or may be omitted.
4. If all parameters are omitted, the sequence number of the first card transferred to the output tape following execution of the SEQ instruction will be assumed as the initial sequence number. Subsequent cards will be sequenced in Columns 5, 6, 7, and 8 and incremented by one.

**Note:** If sequencing is to occur on the right-hand side of the cards (columns 73 through 80), the SEQ control instruction must be preceded by a LOCSEQ instruction specifying RIGHT in the Address and Remarks field.

5. If only the first parameter seqno is specified, seqno will be the sequence number of the first card. Subsequent cards will be sequenced on the last four columns and incremented by one.
6. If only the first two parameters seqno and inc are specified cols will be four, with an increment designated by inc.
7. If the first parameter is omitted, but the remaining two are specified, the Identity and Sequence field of the first card will be assumed.
8. If the first two parameters are omitted and cols is specified, the Identity and Sequence field of the first card will be assumed, and subsequent card sequence numbers will be incremented by one. Columns will be affected as indicated by cols.
9. If the first and third parameters are omitted and only inc is specified, the Identity and Sequence field of the first card will be assumed. Incrementation will be indicated by inc and four columns are to be affected by the sequencing process.

TACSERV  
OUTPUT

Output from TACSERV may consist of an updated TAC language tape and a printed listing of programs written onto the output tape. If a TACSERV control instruction error is detected, the error will be indicated on the Console Typewriter and High-Speed Printer.

Program  
Listing

If LIST is written as a parameter of the ENDALL control instruction, a listing of the ID and block count of each program written on the output tape plus the total block count including sentinels, will be edited and placed on TSYSOUT, edited for printing on the High-Speed Printer.

Console  
Typewriter

Whenever an error is detected during operation of the TACSERV routine, the following indication is typed on the Console Typewriter:

TACS En

where n indicates the type of error as described below:

**High-Speed  
Printer Error  
Indications**

Detection of an error also causes the following indication to be written on the system output tape, edited for printing on the High-Speed Printer:

card contents

\*\*\*\*\* TACSERV Error: Error Statement

where card contents is the illegal card, and error statement is the appropriate error message.

**Error  
Exit**

Following detection and indication of the TACSERV error control is transferred to 1ENDJOB. A listing of all TACSERV errors follows:

<u>Type-Out</u>	<u>Print-Out</u>
TACSE1	ILLEGAL CONTROL CARD
TACSE2	TAPE SPECIFIED IS NEWTAPE OR IS A RESERVED SYSTEM TAPE
TACSE3	NEWTAPE HAS NOT BEEN SPECIFIED
TACSE4	SENTINEL BLOCK PREVENTED COMPLETION OF FUNCTION
TACSE5	END CARD FOUND BEFORE SPECIFIED FLAD/SEQUENCE - NUMBER
TACSE6	ATTEMPTED DELETION OF END CARD
TACSE7	MORE THAN 8 CHARACTERS IN SENTINEL PARAMETER
TACSE8	ILLEGAL PARAMETER IN ENDALL CARD
TACSE9	"I" CARD INTERCEPTED
TACSE10	CARD(S) IGNORED DURING SEARCH FOR "ENDPROG" CARD
TACSE11	ATTEMPTED REPLACEMENT OF AN "END" CARD -- NEW CARD NOT "END"
TACSE12	CARD SPECIFIED BY SYMB PLUS N HAS ALREADY BEEN TRANSFERRED
TACSE13	MORE THAN 8 CHARACTERS IN SEQUENCE PARAMETER

# TACSERV

## TACSERV EXAMPLE

Assume that all programs on Tape OLDTACL, up to but not including program ZEBRA, and all programs on Tape ADDTAPE, up to and including program ABLE, are to be transferred to Tape NEWTACL. Program ABLE is to be revised as it is transferred.

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	NEWTAPE	NEWTACL	\$(1)
	REWIND	OLDTACL, ADDTAPE, NEWTACL	
	COPYTIL	OLDTACL, ZEBRA	\$(2)
	CORRECT	ADDTAPE, ABLE	\$(3)
	LOCSEQ		\$(4)
	SEQ	ABLE0001,,3	\$(5)
	ADD	ABLE0032,1	\$(6)
	TMA	CARD	\$(7)
	ENDPROG		\$(8)
	ENDALL	LIST, BBBBBBBB	\$(9)
			\$(10)

Explanation - Execution of these instructions causes the following action to take place:

1. Tape NEWTACL is assigned as the output tape.
2. Tapes OLDTACL, ADDTAPE, and NEWTACL are rewound.
3. All information prior to program ZEBRA on tape OLDTACL is transferred to tape NEWTACL.
4. All information prior to program ABLE on tape ADDTAPE is transferred to tape NEWTACL, and TACSERV is set to the Class II mode.
5. TACSERV is set to search for cards by sequence number.
6. TACSERV is set to assign sequence numbers to cards within the corrected program ABLE. Sequencing will begin with ABLE0001 and incrementation will be by one.
7. The portion of program ABLE prior to card ABLE0032 is transferred to tape NEWTACL.

8. TMA CARD is written on tape NEWTACL.
9. The remaining portion of program ABLE is transferred to tape NEWTACL and tape ADDTAPE is positioned at the beginning of the block following the last block of program ABLE. TACSERV is set to NOSEQ mode, and reset to accept CLASS I instructions.
10. Two sentinel blocks of B's are written on tape NEWTACL. A listing of the ID and block count of each program written on tape NEWTACL plus total block count is written on TSYSOUT and control is returned to 32KSYS.

## TAPEDUMP

### TAPEDUMP SERVICE ROUTINE

FUNCTION	Edits information from a magnetic tape for printing in various formats.												
SERVICE ROUTINE INITIATION	The TAPEDUMP services routine may be called into operation by the following instruction:												
	<table border="1"><thead><tr><th>L</th><th>Location</th><th>Command</th><th>Address and Remarks</th></tr></thead><tbody><tr><td></td><td></td><td>TAPEDUMP</td><td></td></tr></tbody></table>	L	Location	Command	Address and Remarks			TAPEDUMP					
L	Location	Command	Address and Remarks										
		TAPEDUMP											
INPUT-OUTPUT SYSTEM	TAPEDUMP uses 32KSYS XORD for all tape movements.												
CONTROL INSTRUCTIONS	TAPEDUMP accepts five control instructions written in standard 32KSYS control line format:												
	<table><thead><tr><th><u>Instruction</u></th><th><u>Function</u></th></tr></thead><tbody><tr><td>PROCESS</td><td>Processes the designated tape, converting and editing the data as specified and writing the output on TSYSOUT.</td></tr><tr><td>REWIND</td><td>Rewinds one or more tapes. The function is as described in Tape Handling.</td></tr><tr><td>SPACEF</td><td>Spaces forward on the designated tape.</td></tr><tr><td>SPACEB</td><td>Spaces backward on the designated tape.</td></tr><tr><td>ENDALL</td><td>Terminates the action of the TAPEDUMP routine and returns control to 32KSYS.</td></tr></tbody></table>	<u>Instruction</u>	<u>Function</u>	PROCESS	Processes the designated tape, converting and editing the data as specified and writing the output on TSYSOUT.	REWIND	Rewinds one or more tapes. The function is as described in Tape Handling.	SPACEF	Spaces forward on the designated tape.	SPACEB	Spaces backward on the designated tape.	ENDALL	Terminates the action of the TAPEDUMP routine and returns control to 32KSYS.
<u>Instruction</u>	<u>Function</u>												
PROCESS	Processes the designated tape, converting and editing the data as specified and writing the output on TSYSOUT.												
REWIND	Rewinds one or more tapes. The function is as described in Tape Handling.												
SPACEF	Spaces forward on the designated tape.												
SPACEB	Spaces backward on the designated tape.												
ENDALL	Terminates the action of the TAPEDUMP routine and returns control to 32KSYS.												
PROCESS													
Format	<table border="1"><thead><tr><th>L</th><th>Location</th><th>Command</th><th>Address and Remarks</th></tr></thead><tbody><tr><td></td><td></td><td>PROCESS PROCESS</td><td><u>t, n, f</u> <u>t, sent, f</u></td></tr></tbody></table>	L	Location	Command	Address and Remarks			PROCESS PROCESS	<u>t, n, f</u> <u>t, sent, f</u>				
L	Location	Command	Address and Remarks										
		PROCESS PROCESS	<u>t, n, f</u> <u>t, sent, f</u>										
Parameters	<p><u>t</u> - The symbolic tape except TSYSDMP, TSYSOUT, and TSYSIN, to be processed.</p> <p><u>n</u> - The number (decimal) of blocks to be processed.</p>												

sent - the sentinel word to be used in locating a sentinel block.

f - An alphabetic character that specifies the format in which the tape is to be edited.

A = Alphanumeric

H = Hexadecimal

O = Octal

F = Floating-point

nS = Fixed-point, where n is a decimal number from 0 through 47, which indicates the position of the binary point. If n is omitted, 0 will be assumed.

#### SPACEF

Format	L	Location	Command	Address and Remarks
			SPACEF SPACEF	<u>t</u> <u>n</u> <u>t</u> , <u>sent</u>

#### Parameters

See PROCESS

#### Remarks

If sent is specified, the tape will be positioned immediately following the sentinel block.

#### SPACEB

Format	L	Location	Command	Address and Remarks
			SPACEB SPACEB	<u>t</u> , <u>n</u> <u>t</u> , <u>sent</u>

#### Parameters

See PROCESS

#### Remarks

If sent is specified, the tape will be positioned immediately following the sentinel block (in reference to a backward direction).

## TAPEDUMP

ENDALL

Format

L	Location	Command	Address and Remarks
		ENDALL	

Parameters

None

Action

TAPEDUMP is terminated and control is returned via  
`SYS. 1NXTCOM

## **SECTION XV**

### **APPLICATIONS**

This section deals with applications which run under control of the 32KSYS monitor. Information relevant to use of the application system within the monitor is included; detailed information as to the input and output formats, options, etc., can be found in the documents referenced by each application.

## PERT

### PERT - Performance Evaluation and Review Technique

#### FUNCTION

Initializes a PERT run.

#### FORMAT

L	Location	Command	Address and Remarks
	<u>run</u>	PERT III	<u>t<sub>1</sub></u> , <u>t<sub>2</sub></u> , <u>n</u>

#### PARAMETERS

run - Signifies an initial run or an update run to an existing Master File. One of the following two parameters must be written in the location field:

- INITIAL - an initial run
- UPDATE - an update run to an existing Master File.

t<sub>1</sub> (OPTIONAL) - The symbolic tape indicating the existing Master File. If not present, run must be INITIAL.

t<sub>2</sub> - The symbolic tape indicating the new Master File.

n - The number (decimal) of tape transports available to PERT III. If n is unspecified or is greater than 10, PERT III will utilize up to eleven tape transports. If n = 10 and an old master file tape is specified, PERT III will rewind TSYSCR7 to provide a transport on which to mount a file tape. If n is less than ten an error typeout will ensue and the run will be terminated.

#### ACTION

- The input deck is read, sorted, and merged (if an UPDATE run) with the existing Master File to create a new Master File tape.
- PERT computations are performed.
- Output reports are generated.

#### REFERENCE

The program is described in TM-30, Philco 2000 PERT III System.

**REMARKS**

1. If a PERT III run is successfully completed, the program returns to the system via SYS.1NXTCOM. Otherwise, control is returned via SYS.1ERRDMP.

**CONSOLE  
TYPEWRITER  
NORMAL  
TYPE-OUTS**

<u>Message</u>	<u>Explanation</u>
PERT III	Start of run
MASTER FILE ON T XX	Location of new Master File

**CONSOLE  
TYPEWRITER  
ERROR  
TYPE-OUTS**

<u>Message</u>	<u>Explanation</u>	<u>Action</u>
DUMPPERT	Machine or pro- gram error. Computer halts.	Dump memory and contact Philco Program- ming R & D Dept.
TOO MANY PREDECES- SORS OR SUCCESSORS	Input data error	None. Returns to SYS.1ERRDMP
BAD CONTROL CARD	Illegal or missing parameter(s) in PERT III control card.	None. Returns to SYS.1ERRDMP

**STAT/2000****STAT/2000****FUNCTION**

Initiates a STAT/2000 run.

**FORMAT**

L	Location	Command	Address and Remarks
		STAT	$t_1$ (S), $t_2$ (S)

**PARAMETERS**

$t_1$  - The symbolic tape to be used for storage of data values. If omitted, TSYSCR3 is assigned to  $t_1$ .

$t_2$  - The symbolic tape to be used for storage of data results. If omitted, TSYSCR6 is assigned to  $t_2$ .

(S) (OPTIONAL) - Indicates that  $t_1$  and/or  $t_2$  is to be rewound with lockout. This allows the tapes to be saved for future use on NODATA or FOLLOW option.  
(Refer to Philco Manual TM-36.)

- . The STAT/2000 System is loaded into core.
- . The input deck is read and the requested statistical function is performed.
- . Both printer and plotter output are placed on tape TSYSOUT.
- . Subsequent statistical functions are loaded and executed until an ENDSTAT control card is encountered.
- .  $t_1$  and  $t_2$  are handled as specified by the STAT card, i.e., they are rewound with lockout if (S) has been requested.

**REMARKS**

The use of (S) eliminates the ability to stack runs. When additional statistical functions are desired, the STAT control instruction must be resubmitted releasing tapes  $t_1$  and  $t_2$ .

**REFERENCE**

STAT/2000 control cards, individual statistical programs, and console typewriter typeouts are described in the Philco Manual TM-36.

**XMAS - Expandable Machine Accounting System****FUNCTION** Initiates an XMAS run.**FORMAT**

L	Location	Command	Address and Remarks
		XMAS	

**PARAMETERS** None**ACTION**

- Loads the XMAS Monitor program and transfers control to it.
- The XMAS Monitor then reads the parameter cards following the XMAS control instruction and calls on the XMAS programs required for calculating, collating, report writing, selecting or sorting 80 character card records.

**REFERENCE**

The status of XMAS tapes, the programming language, and console typewriter typeouts are described by TM-28, Philco 2000 XMAS System Manual.

**LP****LP - Linear Programming****FUNCTION** Initiates an LP run.

FORMAT	L	Location	Command	Address and Remarks
			LP	

**ACTION**

- LP reads the input data and subsequent control cards following the LP control instruction.

- LP computations are performed.

- Printer output is placed on tape TSYSOUT.

**REMARKS**

1. TSYSCR0 is used as the scratch for History Tape.
2. If an LP run is successfully completed, the program returns to the system via SYS.1NXTCON. Otherwise, control is returned via SYS.1ERRDMP.

**REFERENCE**

The program is described in TM-7, Philco 2000 Linear Programming System.

## **SECTION XVI**

### **TAC LIBRARY ROUTINES**

Associated with 32KSYS are several subroutines and generators available for placement on the TAC library tape (TSYSLIB). These subroutines and generators have application only within programs run under control of 32KSYS.

32KSYS generators such as SNAPGEN and LOADGEN, provide at compilation time machine language for specific 32KSYS functions.

32KSYS generators are called into operation using the call word in the Command field and the parameters in the Address and Remarks field.

# LOADGEN

## LOADGEN - Loading Generator

### FUNCTION

Generates machine coding at compilation time, which, when executed, causes a program to be loaded via the internal loader. LOADGEN enables a program to be loaded during the running of another program.

### FORMAT

L	Location	Command	Address and Remarks
		LOADRPL or LOADABS	<u>n</u> , <u>id</u> , <u>go</u> \$

### PARAMETERS

LOADRPL - Indicates that a normal RPL program is to be loaded.

LOADABS - Indicates that an ABS program is to be loaded.

n - The PUN (decimal, 0 through 15), of the tape from which the program is to be loaded. It may also be the symbolic location of a word which contains the PUN in bit positions 18 through 23.

id - The identity of the program to be loaded.

go (OPTIONAL) - If GO is written, control will be transferred to the program just loaded. If omitted, control will be returned to the calling program after program id has been loaded.

### ACTION

During compilation of the object program, TAC replaces the programmer's LOADGEN call with a transfer of control to the coding generated by LOADGEN.

### REMARKS

1. After a program is loaded via LOADGEN, control may be transferred to it, or be returned to the original program.
2. Each error condition that is detected by LOADGEN will generate:
  - a. Coding that includes a JMP SYS.1ERRDMP instruction, causing an automatic error dump at run time.

- b. A comment card, which defines the error condition. This card will not have an asterisk in column 9, therefore causing compilation error.
  - c. The comment card THE REMAINING CHARACTERS IN THIS CALL ARE ..... followed by a comment card(s) with the remaining characters of the call.
3. The 32KSYS internal loader may also be used at run time by a transfer of control to SYS.1INTLD.

## EXAMPLE

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
		LOADRPL	6, SEG1, GO\$

Explanation: Execution of this instruction causes coding to be generated at compilation time, that will load program SEG1, located on PUN 6, into memory at run time in RPL format. Control is transferred to SEG1 immediately after it is loaded. The tape on PUN6 remains positioned at the end of SEG1.

## SNAPGEN

### SNAPGEN - Snapshot Generator

#### FUNCTION

Generates machine coding at compilation time, which, when executed, produces selective snapshot dumps during the running of an object program whenever designated conditions are met. SNAPGEN produces the same snapshot dumps as those produced by the SNAP control instruction.

#### CALL WORDS

The SNAPGEN generator may be called either by a SNAP or a SNAPTOG call word. SNAP requests a dump whenever specified conditions are met; SNAPTOG indicates that the snapshot dump is to take place (depending upon conditions) only if a designated toggle switch is in the ON position during the running of a program.

#### SNAP FORMAT

L	Location	Command	Address and Remarks
		SNAP	<u>format</u> : <u>start</u> : <u>end</u> : <u>cond\$</u>

#### SNAPTOG FORMAT

L	Location	Command	Address and Remarks
		SNAPTOG	<u>tog</u> : <u>format</u> : <u>start</u> : <u>end</u> : <u>cond\$</u>

#### PARAMETERS

All parameters must be separated by semicolons.

tog (for SNAPTOG only) - The number (decimal, 0 through 47), which indicates the toggle switch that must be in the ON position before this particular snapshot dump can occur. The parameter may also be a symbol defined by an ASGN or SAME card at compilation time as a decimal number, 0 through 47.

format - See format parameter of SNAP control instruction.

start - An absolute or symbolic location which indicates the first word to be dumped. Any numeric designation is assumed to be decimal. Any configuration preceded by an M/ is assumed to be octal.

end - An absolute or symbolic location which indicates the last word to be dumped. The same rules for start pertain to end.

cond (OPTIONAL) - See cond parameter of SNAP control instruction. If cond is specified, all numbers are assumed to be decimal. M/designations may only be used within parenthesized expressions. A dollar sign (\$) must follow cond. If cond is omitted, the dump is assumed to be unconditional, and a dollar sign (\$) must follow the end parameter.

**ACTION**

During compilation of the object program, TAC replaces the programmer's SNAPGEN call with coding generated by SNAPGEN. Included in this coding is a snap table which contains the information required to perform the snapshot dump, and the call upon the snapshot routine.

**REMARKS**

1. SNAP generator calls may not be placed under the influence of an RPT, TIO, or SKC instruction.
2. Upon execution of the generated coding, the contents of the JA Register are not saved.
3. There is not limit to the number of calls within a particular program. If an error is detected by SNAPGEN during compilation, coding will be generated, which will enable each call containing an error to be bypassed at run time. For each snap error detected, the error condition will be printed on the Code-Edit, along with the statement, THE REMAINING CHARACTERS IN THIS CALL ARE xx...x, where the x's indicate the remaining characters, if any, in the Address and Remarks field.

**EXAMPLE**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
----------	-----------------	----------------	----------------------------

TMA	2052\$
TMD	2047\$
JAED	FINIS\$
SNAP	A;1094;JOE;5/9\$

## **SNAPGEN**

<u>L</u>	<u>Location</u>	<u>Command</u>	<u>Address and Remarks</u>
	TMA	XRAY\$	
	AMS	GEORGE\$	
	SNAPTOG	1;C;START,1X;START, 3X;A<3210\$	
	.	.	
	.	.	
	.	.	

Explanation: The SNAP call causes an alphanumeric snapshot dump of decimal location 1094 to memory location JOE the fifth through the ninth time the SNAP call is encountered. The SNAPTOG call causes a command dump of memory location START plus the contents of Index Register 1 through memory location START plus the contents of Index Register 3, if Toggle Switch 1 is set to the ON position and if the contents of the A Register are less than 3210.

## APPENDIX A

### 32KSYS OPERATING PROCEDURES

#### INITIALIZATION of 32KSYS

In order to initialize the Philco 32KSYS Operating System, 32KSYS, the operator should perform the following operations:

1. For ease of operation, plug the logical and physical tape units one-to-one.
2. Mount the System Tape on Tape Transport 1 and a scratch tape on Tape Transport 2.

#### Magtape Image Mode:

- a. Set all toggles off.
- b. Read one block from Tape 1 into Memory Location Zero.
- c. Execute a JMPL Zero.
- d. 32KSYS (DATE OF VERSION) is typed on the Console Typewriter.
- e. 32KSYS will proceed automatically.

#### Magtape Code Mode:

- a. Set Toggle 47 on.
- b. Read one block from Tape 1 into Memory Location Zero.
- c. Execute a JMPL Zero.
- d. SET TOGGLS is typed on the Console Typewriter and the computer stalls.

- e. Set Toggle 21 on.
- f. Set Toggle 47 off.
- g. The white light signal of the Console Typewriter comes on and the computer stalls.
- h. Set Toggle 21 off.
- i. Type in normal 32KSYS format:

CONIN                    CODE

**Console Typewriter**

- a. Set Toggle 47 on.
- b. Read one block from Tape 1 into Memory Location Zero.
- c. Execute a JMPL Zero.
- d. SET TOGGLS is typed on the Console Typewriter and the computer stalls.
- e. Set Toggle 21 on.
- f. Set Toggle 47 off.
- g. The white light signal of the Console Typewriter comes on and the computer stalls.
- h. 32KSYS is now set to accept all input via the Console Typewriter (Refer to Special Operator Actions, Toggle 21; also refer to CONTROL LINE FUNCTIONS, Control Line Format.)

**Remarks:** If a 32KSYS or Service Routine control line error occurs while in Console Typewriter Mode, the operator must clear error conditions by typing a JOB control instruction before continuing the function.

## **EXECUTION OF A CLOBDUMP**

In order to execute a 32KSYS catastrophic dump (CLOBDMP), assuming Magtape Code Mode, the operator should perform the following actions:

1. Rewind Tape 1.
2. Write one block from Memory Location Zero onto scratch tape on Tape Transport 2.  
Do not rewind Tape Transport 2.
3. Read one block from Tape 1 into Memory Location Zero.
4. Set Toggle 24 on.
5. Execute a JMPL Zero.
6. CLOBDMP is typed on the Console Typewriter and a clobdump is executed.

## **TAPE ASSIGNMENTS**

Two tapes must be assigned specific tape units; the System Tape on TT1 and a scratch (for use as the intermediate dump tape) on TT2. All other tapes may be mounted on any available transport and, as each is required, the typeout TAPE XXXXXXX = occurs, and the computer stalls in a tight loop. At this time, the position of the toggle (0-15) which corresponds to the IOP plug number associated with TAPE XXXXXXX must be changed. The Console Typewriter then types the plug number and a question mark, and the computer stalls again. If this is the correct plug number, the toggle should be returned to its original position. The Console Typewriter again types the plug number and the System will continue operation. If the plug number is incorrect, change the position of another toggle (0-15) and repeat the above procedure.

## **STANDARD SYSTEM TAPE REQUESTS**

### **TYPE-OUT**

TSYSCR0

TSYSCR3

TSYSCR4

TSYSCR6

TSYSLIB

TSYSIN

TSYSOUT

### **TAPE REQUESTED**

System scratch (write-enabled)

System scratch (write-enabled)

System scratch (write-enabled)

System scratch (write-enabled)

System library

System input

System output (write-enabled)

Other tape names, as required, will be noted on the JOB request form.

## ERROR TOGGLES

The Error Toggles are examined every time a JMPL 1 is executed. If an Error Toggle is set, a message is typed on the Console Typewriter and written on TSYS-OUT. Toggle 46 is examined; if it is not set, dumps and/or snaps are edited for printing. A search is then made for the next JOB instruction unless Toggle 47 is set (refer to Special Operator Actions below).

ERROR TOGGLE	MEANING
45	Unexpected halt
44	Program in loop
43	Deficient operating instructions
42	Time limit for run exceeded
41	Tape error
40	Machine error
39	Job arbitrarily stopped

Inhibit dumps/snaps toggle: Toggle 46.

## SPECIAL OPERATOR ACTIONS

Before each JOB control card is executed and each time a JMPL 1 is executed, Toggle 47 (the Action Toggle) is examined. If it is on, SET TOGGLERS is typed on the Console Typewriter and the computer stalls until the toggle is turned off. Then toggles 16 to 23 are examined and the appropriate action is taken by the System.

TOGGLE	ACTION
23	Tapes specified by toggles 0-15 are freed and nn FREE is typed on the Console Typewriter for each tape released.
21	32KSYS is set to accept <u>all</u> input via the Console Typewriter until toggle 21 is set off or until a CONIN IMAGE or CONIN CODE instruction is encountered.
	Whenever control instructions are entered from the Console Typewriter, a carriage return

TOGGLE	ACTION
	character must end each instruction line. If a typing error should occur, the entire instruction may be re-entered by pressing the STOP CODE Key. In response to this action 32KSYS issues a carriage return and requests a new type-in via the white light signal.
	Refer to Remarks under Console Typewriter mode.
20	WRAPOUT is typed on the Console Typewriter and TSYSOUT is wrapped-up.
19	RERUN is typed on the Console Typewriter and the current job is rerun.
18	WRAPIN is typed on the Console Typewriter and TSYSIN is wrapped-up.
17	System searches backwards for JOB card and types the job ID on the Console Typewriter.
16	System searches forward for JOB card and types the job ID on the Console Typewriter.
<b>REMARKS</b>	If toggle 16 or 17 is used, it must remain set until the desired job is encountered.
<b>JOB ENDINPUT</b>	When JOB ENDINPUT is typed on the Console Typewriter and the computer halts. The operator must select one of the above-mentioned actions (e.g., wrapup TSYSIN, wrapup TSYSOUT, etc.), by setting TOGGLE 47 and executing a JMPL 1.

## APPENDIX B

### ALPHABETICAL LISTING OF CONSOLE TYPEWRITER TYPE-OUTS

<u>Type-Out</u>	<u>Error/Normal</u>	<u>Source</u>	<u>Explanation</u>
1DMPTAC	Error	TAC	Either a machine or a program error was detected during the first pass of TAC. A CLOBDMP should be taken.
2DMPTAC	Error	TAC	Either a machine or a program error was detected during the second pass of TAC. A CLOBDMP should be taken.
32KSYS <u>mmddyy</u>	Normal	32KSYS	The system was just initialized. The type-out indicates the system version.
AIDE E1	Error	AIDE	An illegal control card was detected.
AIDE E2	Error	AIDE	A parameter is missing.
AIDE E3	Error	AIDE	A begin-tape error was detected; no sentinel was found.
AIDE E4	Error	AIDE	More than eight characters were specified in the sentinel parameter.
ANAL E1	Error	ANALYZER	An illegal control card was detected.
ANAL E2	Error	ANALYZER	The tape specified is reserved by the system.
ANAL E3	Error	ANALYZER	Parameters are missing.
ANAL E4	Error	ANALYZER	The specified format is not ABS or RPL.
ANAL E5	Error	ANALYZER	An illegal RPL section word was detected.
ANAL E6	Error	ANALYZER	An illegal ABS card was detected.
ANAL E7	Error	ANALYZER	A checksum error was detected.

<u>Type-Out</u>	<u>Error/Normal</u>	<u>Source</u>	<u>Explanation</u>
ANAL E8	Error	ANALYZER	The end parameter was less than the begin parameter.
<u>nn</u> ASSIGNED TO <u>symb</u>	Normal	32KSYS	Logical unit <u>nn</u> is assigned to symbolic tape <u>symb</u> . (The computer will stall until the operator selects another unit).
<u>symb</u> BAD LIB	Error	TAC	Information found on tape <u>symb</u> is not library data.
<u>nnnn</u> BLKS	Normal	TAC	<u>nnnn</u> blocks were produced in the compiled RPL program.
CLOBDMF	Normal	32KSYS	The catastrophic dump is being executed.
CLOCK FAILURE	Error	32KSYS	A clock order was not accepted.
CMPERRS	Error	TAC FORTRAN	One or more serious compilation errors were detected.
CONTROL CARD ERROR	Error	PERT	An illegal card for PERT run was detected.
<u>nnnn</u> CARDS	Normal	TAC OPAL FORTRAN	<u>nnnn</u> cards were generated..
DATA E1	Error	DATA	An illegal control card was detected.
DATA E2	Error	DATA	The tape specified is reserved by the system.
DATA E3	Error	DATA	An illegal format parameter was detected.
DATA E4	Error	DATA	The input is not image as specified.

<u>Type-Out</u>	<u>Error/Normal</u>	<u>Source</u>	<u>Explanation</u>
DATA E5	Error	DATA	The number of words per card was not specified or is illegal.
DATA E6	Error	DATA	The number of cards per block was not specified or is illegal.
DATA E7	Error	DATA	The (number of words per card) x (cards per block) is illegal.
DATA E8	Error	DATA	More than eight characters are specified in a parameter.
DATA E9	Error	DATA	A parameter is not numeric.
DIFFER- ENCES	Normal	AIDE	The comparison of two magnetic tapes is complete and inequalities were detected.
DUMPPERT	Error	PERT	Either a machine or a program error was detected during PERT. A CLOBDMP should be taken.
EALTAC	Error	ALTAC	A source language error has been detected that would cause generation of an incorrect object program. Control was transferred to SYS.1COMPER.
ECOBOL	Error	COBOL	A source language error has been detected that would cause generation of an incorrect object program. Control was transferred to SYS.1COMPER.
ENDCOMP	Normal	FORTRAN	Compilation is complete without serious error.
EREPOR T	Error	REPORT GENER- ATOR	A source language error has been detected. Control was transferred to SYS.1COMPER.
F4DUMP	Error	FORTRAN	Machine or compiler error. Computer halts. Press advance to wrap up compilation.

<u>Type-Out</u>	<u>Error/Normal</u>	<u>Source</u>	<u>Explanation</u>
GENERR	Error	TAC	The generator is requesting more parameters than were supplied.
ID IS <u>id</u>	Normal	TAC OPAL FORTRAN	The identification of a compiled program is <u>id</u> .
ID IS <u>id</u> <u>nn</u> CARDS	Normal	WRTABS	An ABS program of the designated ID, containing <u>nn</u> cards has been written on TSYSOUT.
INSUFFICIENT MEMORY	Error	OPAL	OPAL requires more memory.
JOB <u>id</u>	Normal	32KSYS	The system has recognized a JOB card. <u>id</u> is the first 16 characters of the Address and Remarks field of the JOB card.
LIBE1 to LIBE47	Error	PLUM	For explanation of these PLUM errors see PLUM description in Service Routines.
<u>mm-dd</u> <u>hh-mm.t</u>	Normal	SYS	A JOB card or CLOCK instruction was executed. <u>mm</u> = month; <u>dd</u> = day; <u>hh</u> = hour; <u>mm</u> = minute; <u>t</u> = tenth of minute.
<u>symb</u> NO LIB	Error	TAC	Tape <u>symb</u> is not a library tape.
NO LIB	Error	COBOL	A COBOL library required by the source program is not mounted on the correct tape.
NO IDENTITY	Error	FORTRAN	First non-blank card did not contain program ID. Compilation continues in normal manner.
OPAL ERROR	Error	OPAL	Machine or compiler error was detected.
OPER 39	Error	32KSYS	The job was arbitrarily stopped; the operator dumped the job.

<u>Type-Out</u>	<u>Error/Normal</u>	<u>Source</u>	<u>Explanation</u>
OPER 40	Error	32KSYS	Machine trouble occurred and the operator dumped the job.
OPER 41	Error	32KSYS	Tape trouble occurred and the operator dumped the job.
OPER 42	Error	32KSYS	The time limit for the run was exceeded, the operator dumped job.
OPER 43	Error	32KSYS	Deficient operating instructions; the operator dumped the job.
OPER 44	Error	32KSYS	The program was in a loop; operator dumped the job.
OPER 45	Error	32KSYS	An unexpected halt occurred; the operator dumped the job.
PERT III	Normal	PERT	Start of PERT run.
PROGRAM TAPE IS <u>t</u>	Normal	OPAL	The object program tape is <u>t</u> .
REINIT	Error	32KSYS	The system was not read into memory and must be reinitialized.
RERUN	Normal	32KSYS	The current JOB is to be rerun by operator choice.
RPLC E1	Error	RPLC	An illegal control card was detected.
RPLC E2	Error	RPLC	The tape specified is NEWTAPE or reserved by system.
RPLC E3	Error	RPLC	NEWTAPE was not specified.
RPLC E4	Error	RPLC	A sentinel block prevented the completion of the function.
RPLC E5	Error	RPLC	The card following the IDCHANGE control is not an I card.

<u>Type-Out</u>	<u>Error/Normal</u>	<u>Source</u>	<u>Explanation</u>
RPLC E6	Error	RPLC	An illegal section word was detected.
RPLC E7	Error	RPLC	An illegal correction card was detected and the card was ignored.
RPLC E8	Error	RPLC	More than eight characters were specified in the sentinel parameter.
RPLC E9	Error	RPLC	The <u>id</u> is missing on the addition or correction of an ABS deck.
RPLC E10	Error	RPLC	A checksum error was detected.
RPLC E11	Error	RPLC	An illegal control card was detected.
RPLC E12	Error	RPLC	An illegal parameter was detected in the ENDALL card.
SEARCH NEXT JOB VIA TOGGLES	Error	32KSYS	The system cannot find the next job. The operator should find the next JOB via the Search toggle (Refer to Special Operator Actions.).
SET TOGGLES	Normal	32KSYS	The action toggle was set by the operator (action is specified by toggles 16-23.).
SYSE1	Error	32KSYS	The program is not on the specified tape.
SYSE2	Error	32KSYS	A Checksum error was detected.
SYSE3	Error	32KSYS	The program origin is in list area.
SYSE4	Error	32KSYS	The program origin is in common area.
SYSE5	Error	32KSYS	Undefined refout(s) are specified.
SYSE6	Error	32KSYS	An illegal card was detected.

<u>Type-Out</u>	<u>Error/Normal</u>	<u>Source</u>	<u>Explanation</u>
SYSE7	Error	32KSYS	There was an overflow in the symbol list.
SYSE8	Error	32KSYS	An illegal section word was detected.
SYSE9	Error	32KSYS	An illegal tape name was specified.
SYSE10	Error	32KSYS	The tape name was previously defined.
SYSE11	Error	32KSYS	An illegal address parameter was specified.
SYSE12	Error	32KSYS	An illegal pun was specified.
SYSE13	Error	32KSYS	Too many parameters were specified.
SYSE14	Error	32KSYS	There was an overflow in the PUN table.
SYSE15	Error	32KSYS	A job card was intercepted.
SYSE16	Error	32KSYS	An unaccountable machine or system error was detected.
SYSE17	Error	32KSYS	An illegal 32KSYS control instruction was detected.
SYSE18	Error	32KSYS	An illegal mnemonic was specified.
SYSE19	Error	32KSYS	An illegal decimal number was specified.
SYSE20	Error	32KSYS	An illegal position for the binary point was specified.
SYSE21	Error	32KSYS	Parameter(s) are missing.
SYSE22	Error	32KSYS	Specified symbol is not in the symbol list.

<u>Type-Out</u>	<u>Error/Normal</u>	<u>Source</u>	<u>Explanation</u>
SYSE23	Error	32KSYS	Illegal from-to parameter was specified.
SYSE24	Error	32KSYS	The parameter is too long.
SYSE25	Error	32KSYS	The common size exceeds memory.
SYSE26	Error	32KSYS	An illegal break character was specified.
SYSE27	Error	32KSYS	An illegal magtape parameter was specified.
SYSE28	Error	32KSYS	The NAME.FLAD specified is too long.
SYSE29	Error	32KSYS	An illegal number of words is to be replaced.
SYSE30	Error	32KSYS	An illegal parameter was specified.
SYSE31	Error	32KSYS	TSYSTEM, TSYSEN, and TSYSLIB cannot be write-enabled.
SYSE32	Error	32KSYS	An illegal prog tape was specified.
SYSE33	Error	32KSYS	An illegal octal number was specified.
SYSE34	Error	32KSYS	Too many blocks were specified to be read.
SYSE35	Error	32KSYS	The format for the internal loader was not specified.
SYSE36	Error	32KSYS	Reading to the sentinel will exceed memory.
SYSE37	Error	32KSYS	No address was specified in location field.
SYSE38	Error	32KSYS	An illegal format was specified.

<u>Type-Out</u>	<u>Error/Normal</u>	<u>Source</u>	<u>Explanation</u>
SYSE39	Error	32KSYS	The new common block length exceeds the old.
SYSE40	Error	32KSYS	The program origin equals zero.
SYSE41	Error	32KSYS	Illegal use of TSYSIN if 32KSYS is in Magtape mode.
SYSE42	Error	32KSYS	An illegal START-END parameter was specified.
SYSE43	Error	32KSYS	The third parameter is not (MASTER).
SYSE44	Error	32KSYS	The master program is not defined.
SYSE45	Error	32KSYS	The segment area is empty.
SYSE46	Error	32KSYS	A symbol defined more than once was referenced.
SYSE47	Error	32KSYS	A referenced symbol was re-defined.
SYSE48	Error	32KSYS	It is illegal to locate on TSYSIN.
SYSE49	Error	32KSYS	The relative CSA exceeds PMAX.
SYSE50	Error	32KSYS	LIB must be used if SUBS or NOSUBS specified.
SYSE51	Error	32KSYS	An illegal library tape was specified.
SYSE52	Error	32KSYS	More than 7 library tapes were specified.
SYSE53	Error	32KSYS	Parameter(s) were specified more than once.
SYSE54	Error	32KSYS	A reference was made to an undefined common block.
SYSE55	Error	32KSYS	There was an overflow in the tape name table.

<u>Type-Out</u>	<u>Error/Normal</u>	<u>Source</u>	<u>Explanation</u>
SYSE56	Error	32KSYS	Parameter <u>n</u> is illegal.
SYSE57	Error	32KSYS	Parameter <u>n</u> is missing.
SYSE58	Error	32KSYS	Illegal CSA while loading.
TABOVER	Error	ALTAC3X	There was overflow in the TAC symbol table. Compilation ends.
TACS E1	Error	TACSERV	An illegal control card was detected.
TACS E2	Error	TACSERV	The tape specified is NEWTAPE or is a reserved System tape.
TACS E3	Error	TACSERV	NEWTAPE has not been specified.
TACS E4	Error	TACSERV	A sentinel block prevented completion of a function.
TACS E5	Error	TACSERV	An END card was found before the specified FLAD/sequence number.
TACS E6	Error	TACSERV	Deletion of an END card was attempted.
TACS E7	Error	TACSERV	More than 8 characters were specified in a sentinel parameter.
TACS E8	Error	TACSERV	An illegal parameter was specified in the ENDALL CARD.
TACS E9	Error	TACSERV	AN "I" card was intercepted.
TACS E10	Error	TACSERV	Card(s) ignored during search for "ENDPROG" card.
TACS E11	Error	TACSERV	Replacement of an "END" card was attempted -- new card not "END".
TACS E12	Error	TACSERV	The card specified by <u>symb + n</u> has already been transferred.

<u>Type-Out</u>	<u>Error/Normal</u>	<u>Source</u>	<u>Explanation</u> <sup>*</sup>
TACS E13	Error	TACSERV	More than eight characters were specified in the sequence parameter.
<u>symb</u> TAPE <u>nn</u> IN LOCAL	Error	32KSYS XORD	Unit is mechanically unavailable. Place tape in remote and press advance.
<u>symb</u> TAPE <u>nn</u> WRTNABL	Error	32KSYS XORD	Unit should be write enabled. Write enable the tape and press advance.
<u>symb</u> TAPE <u>nn</u> ROCKED	Error	32KSYS XORD	XORD has made repeated unsuccessful attempts to fix parity and/or sprocket errors. Press advance to retry, or dump job.*
<u>symb</u> TAPE <u>nn</u> ILL. WRT	Error	32KSYS XORD	XORD has been told to write on a tape that has not been write-enabled, and should not be used as an output tape. Dump job.
<u>symb</u> TAPE <u>nn</u> PROGERR	Error	32KSYS XORD	XORD has received, from the calling program, an improper parameter. Dump job.
<u>symb</u> TAPE <u>nn</u> SYSERR	Error	32KSYS XORD	XORD has been partially clobbered, or hardware problem. Dump job.
<u>symb</u> TAPE <u>nn</u> DIS- ABLED	Error	32KSYS XORD	Unit became mechanically disabled while executing an order. Dump job. *
<u>symb</u> TAPE <u>nn</u> BEG TAPE	Error	32KSYS XORD	XORD has been told to process a block prior to the first block on tape. Dump job.
<u>symb</u> TAPE <u>nn</u> END TAPE	Error	32KSYS XORD	XORD has been told to process a block after the last block on tape. Dump job.

\* If serious tape error occurs during compilation, see Remark 2 under TAC.

<u>Type-Out</u>	<u>Error/Normal</u>	<u>Source</u>	<u>Explanation *</u>
<u>symb</u> TAPE <u>nn</u> S1 ERROR	Error	32KSYS XORD	A begin block mark has been missed. Dump job. *
<u>symb</u> TAPE <u>nn</u> S2 ERROR	Error	32KSYS XORD	An end block mark has been missed. Dump job. *
<u>symb</u> TAPE <u>nn</u> BAD FIX	Error	32KSYS XORD	A serious error has occurred while XORD was trying to fix parity and/or sprocket errors. Dump job.
<u>symb</u> TAPE <u>nn</u> FAULTY	Error	32KSYS XORD	Some combination of faults has occurred or there is hardware problems. Dump job. *
TOO MANY PREDEC- ESSORS OR SUC- CESSORS	Error	PERT	An error was detected in the input data.
<u>id</u> TOTAL BLOCKS <u>nn</u>	Normal	FLID	ID of a program and the number of blocks in that program.
TOTAL BLOCKS COPIES <u>nnnn</u> INCLUDING A SENTINEL of <u>xxxxxxxx</u>	Normal	AIDE	<u>nnnn</u> blocks were copied to sentinel <u>xxxxxxxx</u>
WRAPIN	Normal	32KSYS	The wrapup function for tape TSY SIN was executed.
WRAPOUT	Normal	32KSYS	The wrapup function for tape TSY SOUT was executed.

\* If serious tape error occurs during compilation, see Remark 2 under TAC.